Joaquim António Saraiva Patriarca

O Software Livre e de Código Aberto na Administração Pública – Dos mitos às questões de natureza legal, ética e de optimização de recursos públicos

Dissertação de Mestrado em Tecnologias de Informação Geográfica – Ambiente e Ordenamento do Território, orientada pelo Professor Doutor José Gomes dos Santos e apresentada ao Departamento de Geografia e Turismo da Faculdade de Letras da Universidade de Coimbra e à Faculdade de Ciências e Tecnologias da Universidade de Coimbra

2016

· U C ·

Universidade de Coimbra

Joaquim António Saraiva Patriarca

O Software Livre e de Código Aberto na Administração Pública – Dos mitos às questões de natureza legal, ética e de optimização de recursos públicos

Dissertação de Mestrado em Tecnologias de Informação Geográfica – Ambiente e Ordenamento do Território, orientada pelo Professor Doutor José Gomes dos Santos e apresentada ao Departamento de Geografia e Turismo da Faculdade de Letras da Universidade de Coimbra e à Faculdade de Ciências e Tecnologias da Universidade de Coimbra

2016

· U C ·

UNIVERSIDADE DE COIMBRA

Ficha Técnica:

Tipo de trabalho	**Dissertação de mestrado**
Título	**O Software Livre e de Código Aberto na Administração Pública** **Dos mitos às questões de natureza legal, ética e de optimização de recursos públicos**
Autor/a	**Joaquim António Saraiva Patriarca**
Orientador/a	**Doutor José Gomes dos Santos**
Júri	**Presidente: Doutor José Paulo Elvas Duarte de Almeida** **Vogais:** **1. Doutor José António Tenedório** **2. Doutor José Gomes dos Santos**
Identificação do Curso	**2º Ciclo em Tecnologias de Informação Geográfica**
Área científica	**Tecnologias de Informação Geográfica**
Especialidade/Ramo	**Ambiente e Ordenamento do Território**
Data da defesa	**19-2-2016**
Classificação	**20 valores**

· U C ·

UNIVERSIDADE DE COIMBRA

AGRADECIMENTOS

Aos meus Pais e Irmão, berço da minha identidade e fontes inspiradoras da minha personalidade, pelo afecto, amizade e apoio constante, mas, também pelos ensinamentos que me ajudam a crescer como Homem.

À Carolina, pela amizade, apoio, paciência e motivação que sempre me soube transmitir.

Ao Professor Doutor José Gomes dos Santos pela disponibilidade para aceitar a orientação desta dissertação, pela preocupação, dedicação e amizade com que me distinguiu.

Ao Paulo, meu irmão emprestado, por estar presente em todos os momentos da minha vida académica e recente.

A todos os Professores e Colegas que me acompanharam ao longo do meu percurso académico, pelas lições (do Saber e da Vida) que me proporcionaram aprendizagens que deixaram marcas indeléveis na minha personalidade e carácter.

A todos os que, directa ou indirectamente, sempre me apoiaram e contribuíram para vencer os obstáculos e nos permitiram seguir em frente.

Bem-hajam!

O Autor escreve de acordo com as normas vigentes antes do Novo Acordo Ortográfico!

RESUMO

Entre 2008 e 2015, o Estado Português despendeu mais de 37 milhões de euros apenas com a aquisição/renovação de licenças de *Software* Proprietário (SP) SIG, valor que ultrapassa os 250 milhões de euros se lhe adicionarmos os gastos com *software* para sistema operativo (nos quais se incluem programas de produtividade, de escritório, gestão de dados, multimédia, etc.). A tomada de consciência destas avultadas despesas, a crise económica e financeira que o país atravessa e a existência de soluções alternativas que, em princípio, poderão permitir uma gestão mais eficiente da máquina administrativa do Estado Português neste domínio estarão, por certo, na origem da implementação de estratégias e reformas que visam a redução das despesas, impondo a utilização de *Software* Livre e de Código Aberto (SL/CA) quando existe essa possibilidade e quando o seu custo for inferior à alternativa em SP.

O actual cenário legislativo levou-nos a desenvolver uma investigação na qual explanamos, de modo tão esclarecedor quanto julgámos possível, sobre os seguintes assuntos:

i. Prestar esclarecimentos sobre o que é, afinal, o SL/CA SIG e por que razões este deve ser prioridade para a Administração Pública Portuguesa (APP).

ii. Propor estratégias para seleccionar e testar SL/CA SIG, com recurso à execução de exercícios específicos em SIG enquadrados nas competências da APP (modelação da susceptibilidade à ocorrência de movimentos de vertente; modelação da acessibilidade física relativa a infra-estruturas de saúde), com o intuito de comprovar se o SL/CA é uma solução credível e robusta na resolução de problemas espaciais, de modo a competir com a principal referência do mercado constituindo uma verdadeira alternativa a esse *software* modelo.

iii. Ensaiar a génese de uma plataforma (SIGósAPp) para apoio à decisão na selecção do *software* SIG mais adequado à resolução de problemas e projectos específicos, estando nós conscientes de que os organismos públicos estão legalmente obrigados a justificar as suas opções no que diz respeito à selecção/aquisição/implementação de *software*, e sabendo que em alguns casos o SL/CA é a solução mais rentável. Esta plataforma

(aplicação/programa/interface) visa constituir-se como instrumento de apoio aos administradores públicos no momento de decidir sobre a migração para SL/CA, tendo em conta os diversos, difusos e muito definidos e estruturados parâmetros que devem ser tomados em linha de conta no balanço entre o custo toral da solução e o retorno do investimento que, entre outros factores, depende das características das suas especificações, dos objectivos e características da organização/departamento/serviço, necessidade de formação do pessoal técnico, entre muitos outros.

Palavras-chave: *Software* Livre e de Código Aberto, Sistemas de Informação Geográfica, Adopção de *Software* Livre e de Código Aberto SIG na Administração Pública Portuguesa, Avaliação comparada de *software* SIG, Optimização de recursos públicos, SIGósAPp

ABSTRACT

Between 2008 and 2015, the Portuguese State has set aside more than 37 million euros with the acquisition/renovation of GIS proprietary software (PS) alone, a number that exceeds the 250 million landmark if we add to it the expenses with operative system software (in which are included programs related to productivity, office, data management, multimedia, etc.).

Raising awareness of these exacerbated expenses, the financial and economic crisis that the country is currently enduring and the existence of alternative solutions that may enable, presumably, a more efficient management of the administrative machinery of the Portuguese State in this domain area, surely, the cause of spawn for implementing of strategies and reforms targeting expense reduction, imposing the use of Free and Open Source Software (FOSS) when that possibility is in sight and when its cost is inferior to the PS alternative.

The current legal scenario led to the development of an investigation in which we stress out issues, as clearly as we believe possible, upon the following subjects:

i. Clarify the notion of GIS FOSS and for what reasons it must be priority for the Portuguese Public Administration (PPA).

ii. Propose strategies to select and test GIS FOSS, resorting to specific GIS exercises compatible to the PPA field of acting (susceptibility modelling to the occurrence of landslides; physical access modelling related to health-care infrastructures), with the declared purpose of verifying the credibility and strength of the FOSS as a solution for spatial problems, as to compete with the main market reference by constituting itself as a real alternative to that particular software model.

iii. Rehearse the creation of a platform (SIGósAPp) for decision backup in selecting the GIS software most suitable to solving specific problems and projects, while in awareness that public governing bodies are legally obliged to justify their options concerning the software selection/acquisition/implementation, and knowing that, in some cases, FOSS is the most profitable solution. This platform (application, program,

interface) will be depicted as a support instrument for public administrators in the call for decision about migration to FOSS, considering the diverse, diffused and very defined and structured parameters that should be taken in account in the balance between the total cost of ownership and return of investment that, among other factors, highly depends on the characteristics of their specifications, the objectives and traits of the organization/department/service, necessity in training technical staff, and many others.

Key-words: Free and Open Source Software, Geographic Information Systems; Adoption of GIS Free and Open Source Software in Portuguese Public Administration, Comparative assessment of GIS software, Optimization of public resources, SIGósAPp.

AFL – *Academic Free License*

AMA – Agência de Modernização Administrativa

AMC – Análise Multicritério

ANPC – Associação Nacional Protecção Civil

ANSOL – Associação Nacional para o Software Livre

APA – Agência Portuguesa do Ambiente

API – *Application Programming Interface*

APP - Administração Pública Portuguesa

APSDI – Associação para a Promoção e Desenvolvimento da Sociedade da Informação

AUC – *Area Under the Curve*

BD – Base de Dados

BSD – *Berkeley Software Distribution*

CAD – *Computer Aided Design*

CAOP – Carta Administrativa Oficial de Portugal

CCDR – Comissão de Coordenação e Desenvolvimento Regional

COM – *Component Object Model*

COS – Carta de Uso e Ocupação do Solo

CRIF – Carta de Risco de Incêndio Florestal

DGADR – Direcção Geral da Agricultura e Desenvolvimento Rural

DGT – Direcção Geral do Território

DR – Detecção Remota

ESOP – Associação de Empresas de Software Open Source Portuguesas

ESRI – *Environmental Systems Research Institute*

FSF – *Free Software Foundation*

GDAL/OGR – *Geospatial Data Abstraction Library*

GML – *Geography Markup Language*

GNU – GNU *is not Unix*

GPL – *General Public License*

GPTIC – Grupo de Projecto para as Tecnologias de Informação e Comunicação

GRASS – *Geographic Resources Analysis Support System*

IAP – Interoperabilidade na Administração Pública

ICNF – Instituto de Conservação da Natureza e Floresta

IDE – Infra-estruturas de Dados Espaciais

IDW – *Inverse Distance Weighting*

IGEOE – Instituto Geográfico do Exército

IHS – *Intensity, Hue, Saturation*

INSA – Instituto Nacional de Saúde Doutor Ricardo Jorge

INSPIRE – *Infrastructure for Spatial Information in the European Community*

ISA – *Interoperability Solutions for European Public Administrations*

ISO – *International Organization for Standardization*

IUTIC – Inquérito à utilização das Tecnologias da Informação e Comunicação na Administração Pública Central, Regional e Câmaras Municipais

LGPL – GNU *Lesser/Library General Public License*

LNEG – Laboratório Nacional de Energia e Geologia

MDS4OSS – Guia metodológico de apoio à decisão e condução de um processo efectivo de migração, do inglês *Migration Decision Support for OSS*

MDT – Modelo Digital do Terreno

ME-A – Modelo Entidade-Associação, do inglês *Entity-Relationship Model*

MIT – *Massachusetts Institute of Tecnology*

MPL – *Mozilla Public License*

MR – Modelo Relacional, do inglês *Relational Model*

OE – Orçamento de Estado

OGC – *Open Geospatial Consortium*

OSEPA – *Open Source software usage by European Public Administration*

OSGeo – *Open Source Geospatial Foundation*

OSI – *Open Source Initiative*

OSOR – *Open Source Observatory*

OSM – *Open Street Maps*

OSSMETER – *Automated Measurement and Analysis of Open Source Software*

PAH – Processo Analítico de Hierarquização

PGERRTIC – Plano Global Estratégico de Racionalização e Redução dos Custos nas TIC, na administração pública

PME – Plano Municipal de Emergência

PROSE – *Promoting Open Source in European Projects*

RCC – Rede Comum do Conhecimento

ROC – *Receiver Operating Charateristics*

ROI – *Return on Investment*

S4OS – *Software for Operating Systems*

SFLC – *Software Freedom Law Center*

SGBD – Sistema de Gestão de Bases de Dados

SIG – Sistemas de Informação Geográfica

SL/CA – Software Livre e de Código Aberto

SLD – *Styled Layer Descriptor*

RGB – *Red, Green, Blue*

RIT – Rede Irregular de Triângulos

RNID – Regulamento Nacional de Interoperabilidade Digital

SNIT – Sistema Nacional de Informação Territorial

SP – Software Proprietário

TCO – *Total Cost of Ownership*

TI – Tecnologias de Informação

TIG – Tecnologias de Informação Geográfica

VI – Valor Informativo

WFS – *Web Feature Service*

WMS – *Web Map Service*

ÍNDICE GERAL

ÍNDICE DE FIGURAS

- Introdução -

- 1ª Parte -

- 2ª Parte -

- <u>3ª Parte</u> -

ÍNDICE DE TABELAS

- Introdução -

- 1ª Parte -

- 2ª Parte -

- <u>3ª Parte</u> -

INTRODUÇÃO

> *Antes do destino, está a viagem! Como nos disse Pedro Nunes, matemático português que se notabilizou e destacou pela dedicação que emprestou aos problemas subordinados à navegação e aos problemas matemáticos relacionados com uma (Proto) cartografia (náutica) à qual acrescentou o desenvolvimento de instrumentos inovadores (como o Nónio), um navio não conseguiria navegar através das curvas que, ao serem resultantes da intersecção da esfera com um plano secante, correspondem ao caminho mais curto entre dois pontos da superfície esférica (círculos máximos). Ao invés, a embarcação deveria seguir as linhas de rumo ou curvas loxodrómicas, aqueles que intersectam todos os meridianos segundo igual ângulo.*

Pegando nesta importante reflexão, no que diz respeito à definição de uma rota/viagem (aquisição de *software*), face às recentes imposições legislativas, os órgãos que compõem a Administração Pública Portuguesa (APP) necessitam de um rumo, não o mais curto ou rápido, mas aquele que lhe dá garantias de completar o seu trajecto com o maior nível de eficácia, rentabilidade e sustentabilidade.

Para concedermos essa direcção à APP, precisamos (também nós) de estabelecer os marcos da nossa viagem. Então, iniciamos a apresentação desta dissertação com a definição de três aspectos elementares que justificam a sua razoabilidade e delineiam metas mas, antes delas, os caminhos que julgamos mais adequados para as conseguirmos alcançar.

Com o propósito de permitir ao leitor uma mais intuitiva familiarização com os objectivos, metodologias e conteúdos deste documento, optámos por disponibilizar antecipadamente uma brevíssima síntese da sua estrutura, a qual se pode definir por três grandes pilares:

1. **Enquadramento da problemática, motivações e contexto de trabalho**

Trata-se do pilar identitário das nossas motivações pessoais para avançar com o desenvolvimento do documento após a tomada de decisão que nos havia conduzido à escolha da temática em apreço, do qual destacamos os seguintes propósitos:

i. Justificação da escolha do tema e da oportunidade (contexto) da sua abordagem para efeitos de elaboração desta dissertação.

ii. Definir os vectores estruturantes da composição do trabalho, promovendo uma inter-relação entre a APP e as suas necessidades no campo de actuação das Tecnologias de Informação Geográfica (TIG), isto num contexto (ainda algo desconhecido, regulamentado e fiscalizado) que obriga os órgãos que compõem a primeira a justificar as suas opções em matéria de aquisição de *software* e que emana prerrogativas regulamentadas que impõem a implementação de normas e especificações abertas e, naturalmente, se recomenda a adopção de *Software* Livre e/ou de Código Aberto (SL/CA).

2. Objectivos gerais e específicos do trabalho

Trata-se do pilar matricial que anuncia e justifica a ossatura do estudo que aqui se apresenta, no qual nos propomos:

i. Definir os objectivos (gerais e específicos).

ii. Integrar cada um destes objectivos numa estrutura de trabalho lógica e coerente.

3. Abordagem aos aspectos matriciais da metodologia utilizada

Trata-se de justificar e explicar os procedimentos metodológicos adoptados, sempre em contexto de subsídios para propostas de trabalhos futuros e de reflexões críticas, nos quais destacamos alguns procedimentos (propósitos), como:

i. Elencar e caracterizar o espectro (interactivo e dinâmico) de metodologias matriciais utilizadas, tendo em vista a prossecução dos objectivos que nos propomos atingir.

1. Enquadramento da problemática, motivações e contexto do trabalho

> *"Quando falamos em 'software livre'. Estamos a falar de liberdade, não de preço. (…) Distribuir software livre é uma oportunidade de gerar fundos para o desenvolvimento. Não o desperdice!"*
>
> Richard Stallman *apud* Joshua Gay (2002:65) [1]

O presente trabalho surge no seguimento do artigo "Jangada de SIG na Administração Pública Portuguesa" apresentado às I Jornadas de Ciências e Tecnologias de Informação Geográfica, Coimbra, Setembro de 2014 (Patriarca *et al.*, 2014a), uma reflexão que, pelo alcance que registou[2], pelas motivações inerentes e imperativos colocados à APP[3], nos pareceu merecer continuidade.

Com enfoque nos Sistemas de Informação Geográfica (SIG) e tendo por base inúmeras motivações, entre elas as imposições legislativas presentes na letra do Orçamento de Estado (OE)

[1] No original, "*When we speak of 'free software'. We're talking about freedom, not price. (…) Distributing free software is an opportunity to raise funds for development. Don't waste it!" In* GAY, Joshua (2002) – **Free *software*, free society: Selected essays of Richard Stallman**. GNU Press, Boston, 223 p.

[2] Os seguintes endereços URL constituem exemplos do mediatismo que o referido estudo mereceu, ao mesmo tempo que revelam imperativos relacionados com a indispensável redução de custos com a aquisição de *software* e com a formulação de um 'guia de boas práticas':
-» http://www.semanainformatica.xl.pt/administracao-publica/administracao-publica/aberto-mas-nao-muito (acedido em 23 Março, 2015);
-» http://www.computerworld.com.pt/2014/09/02/estudo-confirma-potencial-de-poupanca-com-open-source/ (acedido em 16 de Julho, 2015);
-» http://tek.sapo.pt/noticias/negocios/estado_gastou_24_m_em_*software*_de_informacao_1407609.html (acedido em 16 Julho, 2015);
-» http://www.dn.pt/inicio/portugal/interior.aspx?content_id=4104552 (acedido em 16 Julho, 2015);
-» http://www.publico.pt/tecnologia/noticia/estado-tera-pago-mais-de-100-milhoes-de-euros-por-*software*-que-podia-ter-custo-zero-1668422?page=-1; http://www.rtp.pt/icmblogs/rtp/cientificamente/?m=09&y=2014&d=20 (acedido em 16 Julho, 2015);
-» http://www.ionline.pt/artigos/portugal-tecnologia/estado-portugues-podia-ter-poupado-cinco-anos-100-milhoes-euros-*software* (acedido em 23 Março, 2015).

[3] Entende-se por APP como o conjunto de órgãos, serviços e agentes que, encaixados nas classes taxonómicas definidas pela Lei n.º 3/2004 de 15 de Janeiro e Lei n.º 4/2004 de 15 de Janeiro (Administração directa do Estado, Administração indirecta do Estado e Administração Autónoma), se encontram ligados ao ensino preparatório e superior, serviços centrais (Direcções-Gerais que têm competências em todo o território nacional), serviços periféricos (Direcções Regionais, Embaixadas e Consulados), serviços personalizados (pessoas colectivas de natureza institucional dotadas de personalidade jurídica), fundos personalizados (pessoas colectivas de direito público com natureza patrimonial) e entidades públicas empresariais (pessoas colectivas de natureza empresarial, com fim lucrativo, que visam a prestação de bens ou serviços de interesse público). Enfim, incluem-se, aqui, qualquer organização cuja moldura estatutária a enquadre na estrutura administrativa pública do Estado Português.

de 2013, o referido estudo patenteia resultados deveras interessantes, nomeadamente no que diz respeito ao desempenho de três *software* SIG e aos valores despendidos por alguns sectores da APP com a aquisição/renovação de licenças de *Software* Proprietário (SP).

Porém, um olhar superficial apenas do lado dos gastos imediatos (de leitura directa) com a aquisição/renovação de licenças será sempre periférico, insuficiente e, por essa via, incorrecto. Outras questões se nos colocam[4], entre elas, e num contexto em que, por força do cenário económico internacional e nacional e pelo panorama legislativo em vigor, as instituições da APP, tendo em vista fins de redução de despesa, se encontram obrigadas a equacionar um possível processo de migração para SL/CA, ou pelo menos a justificar todas as opções tomadas no que concerne à contratualização relacionada com a aquisição/renovação de licenças ou serviços ligados a programas informáticos, urge a necessidade de formular um Guia (metodológico) de apoio à decisão e condução de um processo efectivo de migração (adiante designado apenas por MDS4OSS, do inglês *Migration Decision Support for Open Source Software*).

Na medida em que os SIG/TIG não são alheios a estas questões, e visto que a criação de um guia deste tipo é uma tarefa árdua que exige anos de experiência real e empírica sobre a adopção de SL/CA em contexto organizacional, remetemos este documento para um desiderato menos ambicioso, ou seja, a apresentação do conceito de uma plataforma (programa/aplicação) para o apoio à decisão, em matéria de selecção de soluções informáticas SIG/TIG. A SIGósAPp, como a decidimos chamar, é o primeiro contributo para a criação de um MDS4OSS para a APP no âmbito dos SIG/TIG.

Esta resposta que, de modo evidente, a APP necessita, transcende a dimensão financeira. Certamente que as intenções do Governo Português, que legisla a favor do SL/CA por razões, mormente, económicas, constituem uma importante motivação que nos leva a tentar obter esclarecimentos sobre como ajudar os órgãos administrativos públicos. Contudo, não são, nem podem ser, as únicas. Face ao exposto, criou-se um esquema conceptual do trabalho que envolve três vectores estruturantes:

i. APP;

[4] A "Jangada", como primeira reflexão sobre este tema, apresenta limitações e incompletudes, principalmente por não constituir um exercício de comparação directa entre os custos com aquisição/renovação de licenças de *software* comercial e os custos de um processo de migração para *software* livre ou de código aberto, em simultâneo, os testes de desempenho realizados não permitem avaliar globalmente (ou para um problema em específico, os diferentes programas sobre escrutínio, pois debruçaram-se em tarefas elementares, isoladas e descontextualizadas. Grosso modo, estes são alguns espaços vazios que se pretendem ver preenchidos com o desenvolvimento desta dissertação, pondo-a ao serviço das exigências colocadas aos organismos da APP.

ii. *Software* SIG/TIG;

iii. SL/CA.

A Figura 1 expõe esses elementos numa súmula que explora as metas da APP enquanto entidade com competências ao nível do Ordenamento do Território, no âmbito das quais usa imprescindíveis ferramentas para organização, gestão e processamento de dados espaciais (*software* SIG), isto num contexto em que soluções livres e/ou de código aberto podem oferecer o caminho da sustentabilidade, não só financeira, mas também ao nível da eficácia, fiabilidade, sublinhados pelo fortalecimento de valores como a legalidade, ética e transparência.

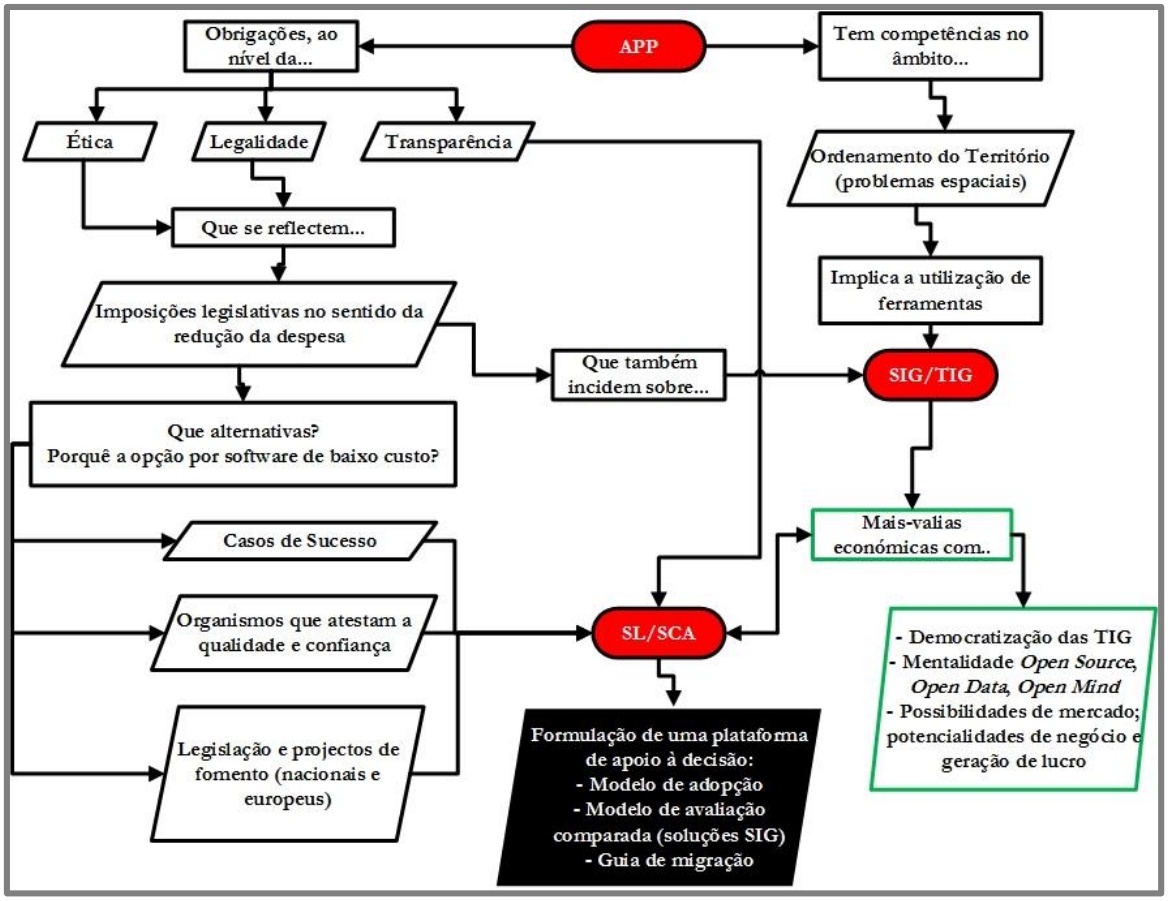


Figura 1 – Vectores estruturantes (e motivacionais) do esquema conceptual da dissertação[5].

[5] As relações expressas nesta figura não podem ser dissociadas do conceito de liberdade, na medida em que a decisão (imparcial) por um tipo de *software*, quer seja SP, SL ou SCA (todos eles, modelos comerciais viáveis), tem de ser obtida de modo consciente e livre, o que pressupõe a expugnação dos interesses de minorias e de quaisquer *lobbies* instalados.

As motivações, digamos o *leit motiv* que nos motivou ao desenvolvimento desta dissertação partiu da ambição e da curiosidade de averiguar a existência de uma reposta para uma questão essencial: 'Porque deverá a APP considerar a implementação de SL/CA SIG?'.

I.1. As Tenologias de Informação Geográfica na Administração Pública Portuguesa: a eterna demanda do Ordenamento do Território

As TIG, no geral, e os SIG, em específico, são ferramentas centrais para a gestão do território e de todas as actividades com uma componente espacial (Cosme, 2012)[6]. No universo da APP são vários os organismos que têm como principal proficiência o planeamento e o Ordenamento do Território, portanto, as TIG são recursos proeminentes, tanto que a relação entre estas e a APP está consagrada na legislação portuguesa em múltiplos diplomas.

A Tabela I apresenta alguns exemplos de actividades desenvolvidas por várias instituições da APP que implicam o recurso aos SIG/TIG. A sua apresentação tem como propósito atribuir razoabilidade à realização da presente dissertação, na medida em que este tipo de ferramenta é essencial para a resolução de um largo conjunto de problemas de índole pública.

Enfim, tendo em conta a importância dos SIG/TIG, perante o dilema de migrar (ou não?) para o SL/CA parece-nos ajustada a ideia de construção da SIGósAPp, uma plataforma que permite constituir-se como auxiliar (em jeito de *handbook*) para apoio à tomada de decisão segundo os melhores interesses de uma organização, mas em conformidade com o enredo legislativo e cumprimento das suas metas.

[6] As TIG e os SIG gerem um amplo espectro de infra-estruturas e tornam solúvel um infinito conjunto de dilemas através da recolha e análise de informação espacial: informação cadastral e gestão de infra-estruturas como redes de transporte (GIS-T), e redes topológicas como as telecomunicações, electricidade, gás e água, de distribuição de bens e produtos e difusão de serviços, serviços de informação cadastral, gestão de resíduos, etc.; planeamento e gestão de recursos humanos e naturais (Ordenamento do Território) – prevenção e actuação em catástrofes, estudos para localização de escolas e de meios (resposta) ou infra-estruturas de saúde, gestão de floresta, elaboração e gestão de um plano director municipal, impacto das alergénicas na saúde humana, etc.; modelação espacial combinada com a dinâmica evolutiva dos fenómenos (inclusão da dimensão temporal) – delimitação das áreas de habitat de uma espécie; propagação de um poluente em cursos de água, etc.; navegação - sistemas de informação para a navegação em navios ou automóveis; produção de cartografia oficial e homologada; aplicações comerciais, não apenas no que diz respeito à venda de *software*, mas também no que concerne à consolidação da venda de outros equipamentos e serviços relacionados com informação geográfica; entre muitas outras.

Tabela I - Exemplos de organismos e competências da APP nas quais é indispensável a utilização de *software* TIG/SIG

Entidade pública	Área temática	Descrição	Legislação de suporte
ANPC[7]	Desastres naturais	Produção de cartografia de risco (inundações; movimentos de vertente, incêndios, etc.); produção de itinerários de evacuação e de retorno.	Lei 27/2006
APA[8]	Planeamento e ordenamento do território	Regula a elaboração e a implementação dos planos de ordenamento da orla costeira (POOC) e estabelece o regime sancionatório aplicável às infracções praticadas na orla costeira, no que respeita ao acesso, circulação e permanência indevidos em áreas interditas e respectiva sinalização.	Decreto-lei 309/93 Decreto-lei 129/2008
Câmaras Municipais	Planeamento e ordenamento do território	Revisão dos Planos Director Municipal, Planos Municipais de Emergência, Planos de Pormenor ou Planos de Urbanização. Execução do registo predial segundo as normas da DGT.	Decreto Reg. 10/2009 Decreto Reg. 11/2009 Decreto-lei 172/95
CCDR[9]	Actividade industrial	Regime jurídico aplicável à revelação e aproveitamento de massas minerais (pesquisa e exploração). A CCDR é consultada sobre a localização e aprova o PARP.	Decreto-lei 270/2001
CCDR	Cartografia temática de ocupação/uso do solo	Carta de ocupação/uso do solo no âmbito da elaboração dos Planos Regionais de Ordenamento do Território.	Decreto-lei 46/2009
DGADR[10]	Agricultura	Monitorizar o uso e a qualidade da água na agricultura.	Decreto Reg. 32/2012
DGADR	Cartografia temática de ocupação/uso do solo	Sistema de Informação de Regadio	Decreto Reg. 32/2012 Decreto-lei 86/2002
DGT[11]	Cartografia temática de ocupação/uso do solo	Produção da Carta de Ocupação do Solo (COS) e *Corine Land Cover*	Portaria 224/2012
DGT	Incêndios	Cartografia de Risco de Incêndio Florestal – projecto que visa a produção de uma carta de perigosidade/risco de incêndio florestal com cobertura para todo o território continental.	Decreto-lei 156/2004

[7] Associação Nacional de Protecção Civil.
[8] Agência Portuguesa do Ambiente.
[9] Comissão de Coordenação e Desenvolvimento Regional.
[10] Direcção Geral da Agricultura e Desenvolvimento Rural.
[11] Direcção Geral do Território.

DGT	IDE – Sistemas Nacionais	Define os requisitos, as condições e as regras de funcionamento e de utilização da plataforma informática destinada ao envio dos instrumentos de gestão territorial para publicação no Diário da República e para publicação na Direcção-Geral do Ordenamento do Território e Desenvolvimento Urbano. Desenvolvimento, gestão e manutenção da plataforma SNIT[12].	Portaria 245/2011
ICNF[13]	Conservação da Natureza e Biodiversidade	Estabelece o Regime Jurídico da Conservação da Natureza e da Biodiversidade	Decreto-lei 142/2008
IGeoE[14]	Cartografia Base	Estabelece os princípios e as normas a que deve obedecer a produção de cartografia no território nacional e aplica-se toda a cartografia topográfica, temática de base topográfica e hidrográfica, com excepção da cartografia classificada das forças armadas	Decreto-lei 193/95 Decreto-lei 202/2007
INSA[15]	Saúde da população humana	Desenvolver métodos destinados à elaboração de previsões, projecções e cenários sobre o estado de saúde e doença da população no futuro, assim como sobre situações ou eventos que possam vir a afectar de forma potencialmente grave a saúde da população.	Regulamento 329/2013
LNEG[16]	Geologia (cartografia)	Cartografia Geológica, Hidrogeológica e de Recursos Minerais	Portaria 425/2012 Decreto-lei 145/2012
LNEG	Movimentos de massa em vertentes	Realizar estudos geológicos específicos em domínios da especialização, como os necessários à pesquisa do risco e perigosidade geológica. Determina a elaboração do Plano Sectorial de Prevenção e Redução de Riscos.	Delibera. 1495/2013
LNEG	Planeamento e Ordenamento do Território	Estratégia Nacional para os Recursos Geológicos. Sistematização e disponibilização do conhecimento por via presencial e, ou, remota.	Resolução C.M. 78/2012

[12] Sistema Nacional de Informação Territorial.
[13] Instituto de Conservação da Natureza e da Floresta.
[14] Instituto Geográfico do Exército.
[15] Instituto Nacional de Saúde Doutor Ricardo Jorge.
[16] Laboratório Nacional de Energia e Geologia.

I.2. A providência legislativa na expugnação dos excessos do passado

Na linha do raciocínio empregue na Figura 1, devemos salientar que desde 2011/12 se tornou incontornável que os organismos que constituem a APP procedam dentro da legalidade e da razoabilidade ética[17] ao terem de adoptar as normas e especificações abertas e, até como consequência dessa decisão/imposição, passar a utilizar SL/CA.

Neste âmbito, condenam-se todas as actividades ilícitas relacionadas com a pirataria de SP (a nível individual e institucional), ou com concursos públicos que visam a obtenção de *software* de forma ilegal (ou outros procedimentos menos transparentes)[18], achando-se que todos devem ter a mesma oportunidade/liberdade de promover, publicitar e concorrer com os seus serviços, de maneira a fomentar a transparência, a legalidade e o bom (justo) funcionamento do mercado, incentivando a livre concorrência (motor do desenvolvimento económico) e diminuindo a despesa pública (na medida em que se encontram soluções menos custosas). Só assim, com todas as alternativas em cima da mesa, será possível, àqueles investidos de responsabilidades de gestão de organismos públicos, optar, de forma racional, legítima e ponderada, pela solução mais vantajosa, considerando os custos monetários e humanos da mudança ou falta dela.

Por si só, os pressupostos e fundamentos do SL/CA são promotores da ética, da legalidade, da propagação do conhecimento, e, sobretudo, da transparência. O SL/CA, pelo acesso ao código-fonte (e possibilidade de alteração de adaptação às necessidades), é um importante veículo de transmissão e desenvolvimento do conhecimento (e de todas as vantagens associadas), do mesmo modo que permite um *workflow* transparente (Krogh & Spaeth, 2007). O SP, por assentar num conceito de *black box* (fechado, de propriedade exclusiva), não permite um exercício de avaliação do código-fonte das ferramentas[19]. Julgamos que é inequivocamente legítimo exigir-se o acesso ao

[17] Aqui, a ética aparece intrinsecamente combinada com as capacidades de boa governança e gestão, bem como com o bom senso que qualquer administrador investido de responsabilidades públicas deverá ter, contribuindo para o desmantelamento das estruturas que alimentam os "interesses instalados" de vários intervenientes envolvidos no mercado, tantas vezes verdadeiros sorvedores dos recursos públicos – que são de todos nós, e a constituição de modelos sólidos de avaliação que promovam a justa escolha das soluções mais favoráveis.

[18] Segundo os investigadores do Projecto OSEPA, a corrupção é um dos problemas que condiciona a adopção de SL/CA. A Associação de Empresas de *Software* Open Source Portuguesas (ESOP), em nota de imprensa disponível em http://www.esop.pt/Default/pt/Destaques/PressRelease?A=2 (acedido em 15 Março, 2015), vai mais longe, dizendo que, em 2013, identificou 47 procedimentos ilegais para a aquisição de *software*, entre um universo de 363 concursos.

[19] Felizmente, alguns programas *black box* possuem ferramentas de desenvolvimento baseadas no código-fonte nativo, as chamadas *Application Programming Interface* (API), que de algum modo atenuam as restrições de acesso ao código-fonte. Isto torna a discussão da abertura do código pouco pacífica, pois embora, com SP, não haja acesso ao código nativo, é possível perceber o modo de funcionamento das suas ferramentas e, inclusive, integrar procedimentos externos que até podem ser baseados noutras linguagens que não a do *software* em questão.

código-fonte, uma vez que só a sua interpretação permite avaliar a aptidão de determinada ferramenta para um problema específico, garantindo-se, assim, a exactidão matemática do processo e a coerência lógica dos resultados. A abertura do código-fonte solucionará as discussões em torno dos diferentes resultados obtidos com processos e ferramentas análogas.

Em suma, é do interesse público utilizarem-se *software* e ferramentas abertas, só assim se garante a imparcialidade, objectividade, qualidade e rigor da solução apresentada, fazendo com que a tomada de decisão seja condizente com os interesses públicos.

Estes valores servem-nos também de sustente e materializam-se mediante a divulgação de projectos e promulgação de diplomas legislativos que fomentam a implementação de SL/CA na administração pública. Para além do espectro normativo internacional que certifica o SL/CA (de que se falará de seguida) e de outros projectos Europeus que incentivam a sua implementação (OSOR[20]; OSEPA[21], ISA[22], PROSE[23] ou OSSMETER[24]), a recente legislação portuguesa sobre a aquisição de licenças de *software* é peremptória ao definir que apenas se podem adquirir licenças de SP nos casos em que seja fundamentalmente demonstrada a inexistência de soluções em SL/CA ou em que o custo total de utilização de SL/CA seja superior à solução em proprietário ou sujeito a licenciamento específico, incluindo nestes, todos os eventuais custos de manutenção, adaptação, migração ou saída (Patriarca *et al*., 2014a). Isto tenta pôr termo aos gastos exorbitantes com a aquisição/renovação de licenças de SP – só entre 2008 e 2015, o Estado despendeu mais de 200 000 000 €. Face ao exposto, estas medidas são legítimas, lógicas e razoáveis.

Lógica, sobretudo, porque não impele a APP a adoptar forçosamente soluções livres. Sabe-se que o SL/CA não é necessariamente gratuito, embora grande parte dos projectos disponibilize os seus *software* livremente e sem encargos monetários, não deixam de existir custos (não só monetários)[25]. Em alguns casos, a tecnologia de código aberto não tem maturidade suficiente para ser utilizada em contextos como aqueles que se exigem a determinada instituição[26]. Por estes dois motivos, a exigência de uma avaliação comparada de soluções em código aberto e SP, tratando-se de um

[20] *Open Source Observatory and Repository*.
[21] *Open Source Software usage by European Public Administration.*
[22] *Interoperability Solutions for European Public Administrations.*
[23] *Promoting Open Source in European Projects.*
[24] *Automated Measurement and Analysis of Open Source Software.*
[25] Custos relacionados com suporte, com formação, com as competências e o desempenho do funcionário ou com possíveis extensões/ferramentas computacionais extra que se tenham de vir a desenvolver ou comprar. Ainda assim a solução SL/CA é bastante promissora quando o desejo é a redução de despesas, essencialmente a longo prazo, não só pela redução drástica dos valores gastos com a aquisição/renovação de licenças, mas também pelo aumento da longevidade dos equipamentos – uma questão que passa também pelo conceito de interoperabilidade.
[26] Chen *et al.* (2010), Akbari & Rajabi (2013) e Steiniger & Hunter (2013) distinguem diferentes níveis de maturidade nos SL/CA SIG que analisam nos seus trabalhos. Para além disto, dados adquiridos ao longo do desenvolvimento desta dissertação identificaram alguns exemplos de adopção, cujo balanço final não foi positivo.

processo de análise e avaliação de desempenho (*benchmarking*) será, porventura, a opção mais sensata para suporte aos decisores.

1.3. A democratização das Tecnologias de Informação Geográfica impulsionada por uma solução robusta confiável

O SL/CA parece uma boa alternativa para a APP redefinir o seu rumo ainda que se justifiquem as dúvidas plasmadas pela simples questão: 'Mas que garantias de qualidade, fiabilidade e segurança pode este tipo de solução informática (SL/CA) assegurar para regular o funcionamento das instituições da administração pública estatal?'. A mensagem transmitida pelos movimentos SL e SCA (FSF – *Free Software Foundation* e OSI – *Open Source Initiative*, respectivamente) é constantemente deturpada, o que se repercute numa falta de confiança que contribui para uma resistência e inércia cultural, em que as pessoas sentem dificuldade em sair da sua área de conforto, evitando o uso de SL/CA.

Esta desconfiança, reveladora de uma certa inércia e resistência à mudança é, em certa medida, incompreensível, dado que também na área dos SIG e das TIG existem normas, protocolos e especificações internacionalmente aceites, homologadas e protegidas por órgãos que asseguram e avaliam a qualidade deste tipo de *software*, definindo padrões de qualidade dos formatos. Destacam-se, a título de exemplo, a *Open Source Geospatial Foundation* (OSGeo) ou o *Open Geospatial Consortium* (OGC). Para além destes, na área da cartografia, a *International Organization for Standardization* (ISO) tem publicadas variadíssimas normas que determinam modelos, protocolos e formatos de referência e de qualidade, e incidem sobre aspectos particulares de eminente interesse prático para o desenvolvimento de programas e produção de dados (Matos, 2008).

Para além disto, são vários os exemplos de publicações periódicas, nas quais se utilizam ferramentas TIG/SIG livres e de código aberto (consultar, por exemplo, a plataforma http://www.sciencedirect.com/, acedido em 15 Novembro, 2015).

Na mesma medida, são já também conhecidas boas práticas associadas a iniciativas de migração protagonizadas por vários organismos públicos, e que constam nos endereços URL: http://www.*software*livre.gov.pt/boas_praticas/ e http://www.*software*publico.gov.pt/boas-praticas. Para a APP, a partilha de boas práticas é indispensável, pois é a partir da experiência empírica que se podem evitar erros passados e eliminar a falta de confiança das soluções livres.

De algum modo, estes três apontamentos creditam a robustez do SL/CA, que, juntamente com o seu modo de distribuição e com a implementação, em paralelo, de uma política que viabiliza a abertura e disponibilização gratuita de informação, definem, no âmbito dos SIG/TIG, aquilo a que podemos chamar 'Democratização das TIG/SIG'.

O SL/CA, mais do que o impacto que possa vir a ter na redução de custos, e como bem elucidam Krogh & Spaeth (2007), Riehle (2010) e Aparício & Costa (2012), acarreta impactos micro e macroeconómicos, fruto da sua génese comercial (Mahony & Naughton, 2004; Fitzgerald, 2006), e expressos, por exemplo, na aceleração de processos de inovação no tecido empresarial, algo que extravasa a APP (novo fôlego no desenvolvimento da indústria do conhecimento e inovação). Enfim, há retornos que relativizam questões relacionadas com quaisquer custos de implementação.

No que respeita às TIG, a ideia de *open software* aparece inevitavelmente associada a outras duas, *open data* e *open mind*[27]. Estas três ideias são eixos estruturantes do desenvolvimento das TIG e, de um modo geral, das Tecnologias de Informação (TI), na medida em que se traduzem em mais-valias económicas. A democratização de que se fala fornece tecnologia e matéria-prima para inúmeras aplicações comerciais que dinamizam a economia, potenciando as empresas do mercado e outros empreendedores que inventam produtos de índole espacial, beneficiadores do cumprimento das regras que garantem a interoperabilidade entre formatos, que, por seu turno, facilitam a troca, disponibilização e manipulação de informação geoespacial ou outra.

Este último aspecto é fundamental num contexto em que a disponibilização de informação é obrigatória. Para além do sector empresarial, a APP só tem benefícios em operar com tecnologia que segue à risca todas as normas de interoperabilidade.

Por sua vez, a entrada no mercado e a competitividade advinda do crescente uso de SL/CA representa a sua melhoria gradual e contínua, e manifesta-se numa maior capacidade para competir (Bouras *et al.*, 2013).

Em suma, estamos perante uma solução interoperável, aparentemente robusta e confiável, cujos benefícios para a sociedade transcendem a simples redução de custos.

---

[27] A mentalidade *open source software* aparece inevitavelmente associada a outras duas ideias, *open data* e *open mind*. Estas não podem ser descontextualizadas de iniciativas internacionais como a Directiva Europeia INSPIRE (Infra-estrutura de Informação Geográfica da Comunidade Europeia) ou o Programa de Monitorização Terrestre Europeu (Copernicus), que, quando enquadrados nos objectivos da Estratégia Europa 2020, denotam uma postura de produção (segundo formatos e normas abertas definidas por outras organizações internacionais como a ISO e a OGC) e disponibilização livre, dentro de limites aceitáveis de segurança, de conhecimento derivado de informação geográfica europeia, com o objectivo de promover tomadas de decisões conscientes e adequadas à realidade e contribuir para a formação de cidadãos informados e conscientes, bem como fomentar o desenvolvimento económico, resultado da criação, *a jusante*, de serviços inovadores relacionados com a informação geográfica produzida no âmbito destas iniciativas.

Sintetizando os raciocínios e reflexões que procurámos expor, a APP encontra-se como que "à deriva" e dependente de factores externos para que, em cada contexto organizacional se decida por proceder à migração, tendo em vista a optimização dos seus serviços e redução da despesa. Esta convicção, fundamentada no próprio suporte legislativo mais recente, nacional e comunitário, e o edifício de factores enquadrantes tais como:

i. Importância das TIG/SIG na APP;

ii. Imposições legislativas e valores morais que levam a APP a equacionar soluções informáticas alternativas;

iii. Vantagens estruturais do SL/CA no âmbito das TIG,

conferem pertinência à ideia de formulação de uma plataforma que ajude as instituições da APP quanto à avaliação de diferentes soluções informáticas SIG.

Todavia, este é um projecto ambicioso e exigente, para o qual ainda não dispomos de uma resposta rigorosa e adequada, visto que este é um trabalho multidisciplinar que envolve inúmeras áreas do saber, e exige a recolha de informações empíricas e experiências vividas sobre processos de adopção concretizados no âmbito da APP. Ainda assim, motivados por este contexto, anunciamos a intenção de desenvolver um documento de matriz científica, que, primando pelo rigor e imparcialidade, combate a ausência de literatura sobre a definição de critérios orientadores (parâmetros) que auxiliam na tomada de decisão no sentido da adopção de SL/CA em Portugal e sobre as verdadeiras capacidades do SL/CA SIG, focando-se em três pontos essenciais que juntos formam o organograma geral que descreve o fluxo de trabalho seguido bem como a organização do seu esquema conceptual (Figura 2):

i. *A raiz do problema* – O que é, afinal, o SL/CA SIG, e como devemos explanar, de modo esclarecedor, as razões pelas quais deve ser prioridade para a APP – **"Porquê considerar *software* SIG livre e/ou de código aberto?"**

ii. *O corpo do trabalho* – Testar SL/CA SIG em exercícios específicos enquadrados nas competências da APP, de modo a validar a sua robustez e fiabilidade através da resolução de problemas complexos – **"Que perspectivas de utilização na APP?"** Esta é uma das questões centrais do trabalho, na medida em que conduz à prossecução da prova de que existe soluções SIG livres confiáveis, transmite conhecimento, segurança e confiança a quem tem de tomar a decisão sobre a migração ou manutenção das soluções actuais. De outro modo, se admitíssemos por mera hipótese a ineficácia

deste exercício, tal seria motivo dissuasor para implementar a ideia de desenvolver a plataforma SIGósAPp.

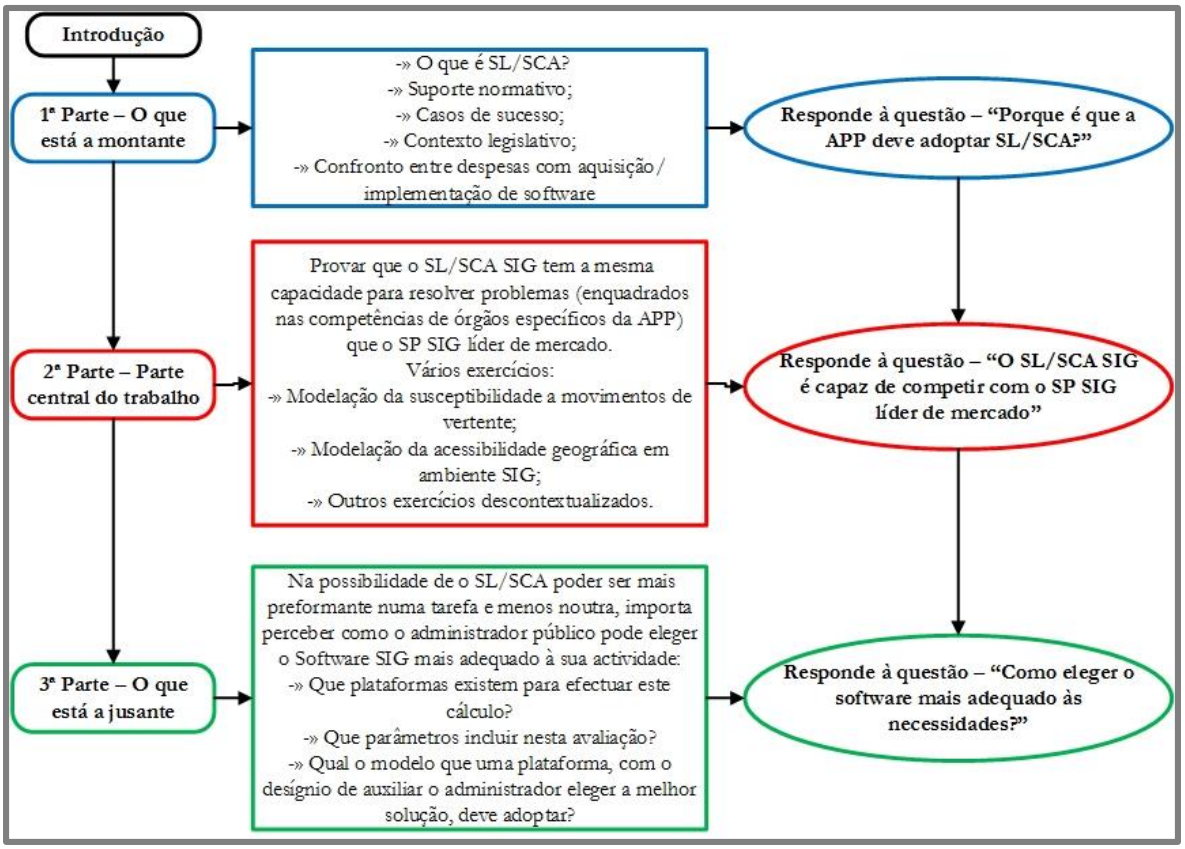


Figura 2 - Organograma geral da dissertação.

i. *Os corolários e as conclusões* – A experiência diz-nos que o SL/CA SIG tem boas performances na resolução de determinados problemas, mas noutros poderá não ser tão eficaz; assim sendo, como é que o administrador público pode saber se determinado *software* SIG é, ou não, o mais adequado para o seu organismo, tendo em conta as actividades que desenvolve e o suporte de apoio de que necessita para a formação dos seus quadros? **"Como seleccionar as melhores soluções informáticas para um organismo público?"** – é a questão que se coloca.

2. Objectivos gerais e específicos do trabalho

> *"Os administradores de instituições públicas têm a missão de dar a melhor direcção aos recursos disponíveis de um modo responsável, transparente e eficaz em termos económicos. O Software Livre e de Código Aberto … oferece aos organismos públicos uma relação custo-benefício, um conjunto de ferramentas reutilizáveis que podem ser um ímpeto à inovação, empreendedorismo e crescimento económico."*
>
> Christos Bouras *et al.* (2014:237)[28]

No primeiro momento de conceptualização e estruturação da presente dissertação tomaram-se como metas matriciais as seguintes:

i. Formular, conceptualizar e apresentar uma plataforma (aplicação/programa/interface), designada por SIGósAPp, para o apoio à decisão (num contexto em que os órgãos da APP são obrigados a justificar as suas opções em termos de selecção de *software*), que permita aos administradores públicos, consoante as suas especificações, objectivos e características da sua organização, eleger soluções informáticas SIG, de um modo o mais automático possível;

ii. Adiantar formulações/ideias sobre a constituição de um guia (metodológico) – designado por MDS4OSS -, por um lado, de apoio à decisão, por outro, de condução em contexto efectivo de migração, no caso de se perceber que uma solução livre e de código aberto seja a mais vantajosa.

No entanto, a dado momento, numa tentativa lograda de as concretizar, apercebemo-nos que estas metas não são alcançáveis no tempo que tínhamos disponível. Tais objectivos implicariam a

[28] No original, "*Public Administrations have the mission of best allocating available resources in a socially responsible, transparent and economically efficient manner. Free and Open Source software (FOSS), being a public resource based on non-rival use rights and allowing for lower entry barriers in software development, offers public stakeholders a set of cost-effective, re-usable tools and resources that can give impetus to innovation, entrepreneurship and economic growth". In* BOURAS, Christos; FILOPOULOS, Anestis; KOKKINOS, Vasileios; MICHALOPOULOS, Sotiris; PAPADOPOULOS, Dimitris & TSELIOU, Georgia (2013) – "Policy recommendations for public administrators on free and open source *software* usage". **Telematics and Informatics**, N.º 31, 237-252.

constituição de uma equipa multidisciplinar, que, na sua orgânica, disponha de conhecimentos em matérias tão díspares como a economia, gestão de empresas e psicologia das organizações e informática, bem como de outras competências técnicas no âmbito dos SIG.

Como se imagina, com excepção das últimas, não dispomos de habilitações de nível académico nas restantes. Para além deste *handicap*, destacam-se outros que nos levaram a repensar os objectivos inicialmente propostos para este trabalho equacionando a sua reformulação num sentido próximo que se apresenta:

i. Um trabalho deste tipo não pode ser concretizado sem o envolvimento de entidades (ou indivíduos) que, estando directamente ligados à APP, disponham de dados empíricos relevantes sobre os factores que condicionam a adopção de SL/CA, nomeadamente SIG, em Portugal[29]. Apesar dos nossos esforços não conseguimos obter toda a informação pretendida – esforços patentes na limitada contribuição adstrita a um inquérito temático que elaborámos, e tivemos oportunidade de disponibilizar e divulgar por um número significativo de órgãos (na expectativa que estes colaborassem também, quer na respostas quer na sua difusão). Foi até com alguma mágoa e decepção que nos apercebemos que maior parte dos inquiridos estão tomados pelo desinteresse e pela falta de conhecimento sobre estas questões.

ii. A esta condicionante junta-se um certo alheamento e até desnorte revelados nos contactos que efectuámos com algumas das instituições que têm responsabilidades de monitorizar, agilizar e regular o processo de justificação das opções de selecção do *software* na APP. Parece-nos claro que estes organismos estão satisfeitos com a plataforma que conceberam para calcular os custos de um *software* e compará-lo com outros, ignorando o facto de que se trata de um modelo genérico que valoriza somente a componente financeira, sendo olvidados ou relegados para um plano de menor destaque factores focados na validação da eficiência e robustez da alternativa, ou aqueles que se preocupam com o impacto que a adopção de uma determinada solução pode ter no tecido organizacional e vice-versa. Esta postura privou-nos de apoio do qual não podíamos prescindir.

Perante estas condicionantes, fomos forçados a redefinir novas metas e objectivos, mais modestos, embora não sejam menos importantes em valor. A avaliação comparada da eficácia e eficiência de

[29] Sem os relatos de quem já enfrentou um processo de migração, explicitando o que correu bem (ou mal) e porque razão correu bem (ou mal), a quantificação de um conjunto de parâmetros envolvidos no modelo de adopção ficará sempre seriamente comprometida.

diversos *software* é um dos elementos mais relevantes no cálculo do seu *Total Cost of Ownership* (TCO), portanto, esta constitui uma das dimensões a integrar na aplicação que visa estabelecer os custos totais da solução informática (quer seja proprietária ou de código aberto). Assim, nesta dissertação, dedicamo-nos sobretudo à experimentação de *software*, tendo como suporte dois argumentos fundamentais:

i. Adquirir dados relativos à performance do maior número possível de ferramentas, tendo em vista a constituição de uma base de dados (BD) que será usada para armazenamento e inquirição destas informações aquando do processo de cálculo da dimensão "custo técnico" do *software* – BD que suportará o funcionamento da SIGósAPp;

ii. Demonstrar que o SL/CA, em SIG tem capacidades que lhe permitem resolver problemas específicos que se colocam à APP.

2.1. Objectivos gerais

Posto isto, com base no que se disse e no esquema motivacional descrito no tópico anterior, adiantam-se agora, no Tabela II, os desideratos gerais da presente dissertação:

Tabela II – Quadro-síntese dos objectivos gerais

	Objectivo geral
1ª Parte	Demonstrar que a APP deve considerar a migração para SL/CA SIG, fazendo bom uso de argumentação relacionada com as recentes imposições legislativas, com o contexto normativo que lhes conferem certificação, com as demonstrações de qualidade expressas em casos de sucesso, e com a comparação entre as despesas com aquisição/renovação de licenças de SP e as despesas com a contratualização de serviços relacionados com SL/CA.
2ª Parte	Comprovar que o SL/CA SIG é uma solução credível e robusta na resolução de problemas espaciais específicos enquadrados nos campos de actuação de órgãos da APP, isto quando comparado ao SP - referências de mercado.
3ª Parte	Formular o conceito de uma plataforma (aplicação/programa/interface) –SIGósAPp -, para o apoio à decisão, permitindo aos administradores públicos, consoante as suas especificações, objectivos e características da sua organização, seleccionar soluções informáticas SIG, do modo mais automático possível.

Estes objectivos matriciais organizam-se segundo o organigrama geral da dissertação (cfr. Figura 2).

2.2. Objectivos específicos

Pela sua abrangência, estes pressupostos gerais desdobram-se em vários objectivos específicos (de suporte à concretização dos primeiros), e estão sistematizados na Tabela III.

Tabela III - Quadro-síntese dos objectivos específicos

	Objectivo
1ª Parte	Clarificar os conceitos de SL e SCA, desmitificando-os de forma científica, imparcial e rigorosa, num esforço que pretende eliminar possíveis confusões que subsistem sobre o que eles implicam realmente.
	Oferecer à comunidade, dados e informações credíveis sobre os problemas relacionados com a migração para SL no seio da APP, contrariando a ausência de literatura e dados empíricos sobre a adopção desse tipo de *software* em Portugal.
	Aferir o grau de confiança do SL em termos de suporte (normativo que garante a qualidade técnica).
	Atestar o grau de confiança de SL/CA através da apresentação de casos de sucesso ou insucesso, nomeadamente nacionais (APP).
	Quantificar as despesas que a APP teve com a aquisição/renovação de licenças de SP e com a contratualização de serviços que envolvem a implementação de sistemas baseados em tecnologia livre e/ou de código aberto, contrapondo as diferenças entre os dois.
2ª Parte	Comparar a performance e o desempenho do SP e SL/CA SIG na resolução de dois processos avançados de modelação espacial (modelação da susceptibilidade a movimentos de vertente e modelação da acessibilidade geográfica) – contemplando a capacidade de processamento com a variação do volume de dados -, de modo a perceber se elas são significativas ao ponto de justificarem a preferência, ainda predominante, pelo SP.
	Apontar possíveis *bugs* ou limitações encontradas nos vários pacotes de *software* testados.
	Estabelecer um grau de complexidade para cada um dos procedimentos de cada um dos *software* em confronto, confirmando ou desmentindo se a utilização de SL/CA está reservada apenas a utilizadores avançados.
	Reflectir de forma crítica, sobre algumas dúvidas relativas a determinados mitos que sugerem que os resultados gerados por ferramentas análogas de diferentes programas não é o mesmo, comparando-se estatisticamente os *outputs* gerados pelos diferentes procedimentos.
	Distinguir os factores (parâmetros) que importam avaliar e ponderar na migração para SL/SCA (nomeadamente os relacionados com a resistência à mudança por parte do pessoal ao serviço).
3ª Parte	Construir um modelo mais objectivo e operacional a partir do TCO/*Return on Investment* (ROI) para efectuar uma análise comparada dos custos associados a cada uma das soluções – proprietária e livre (incluindo custo de migração e da pós-migração), envolvendo factores financeiros, técnicos e humanos (relacionados com o pessoal ao serviço), designadamente em contexto de ajustes directos que envolvam montantes superiores a 10 000 €.
	Explicar as bases, conceptual e lógica, em que assenta a plataforma SIGósAPp.

3. Abordagem aos aspectos matriciais da metodologia utilizada

> *"É comum descrever inferência indutiva como um processo que parte do específico para formular conhecimento e leis sobre o geral, partindo de um número de observações que são usadas para predizer observações futuras que permanecem desconhecidas."*
>
> Pawet Cichosz (2015:4)[30]

Em Ciência, o percurso que nos transporta do incógnito ao saber exige planeamento, um caminho ('método') para o estudo ('logia'). Em si, a explicação dos métodos matriciais utilizados não é tarefa de difícil execução; ainda assim, por se tratar de um trabalho multidisciplinar, o tratamento de cada uma das suas secções obriga à articulação entre vários métodos, ou, em sentido restrito apenas técnicas e processos, tão divergentes quanto os temas que aqui se combinam e relacionam.

Tentámos, por isso, organizar esta informação multifacetada numa exposição que facilite a compreensão da pertinência que a sua inclusão tem na estrutura do presente documento. A Figura 3 sistematiza os vários métodos usados, discriminando grupos agregados segundo as suas propriedades, objectos de interesse e enquadramento na estrutura da dissertação. Distinguem-se, então, quatro grandes grupos:

i. Revisão bibliográfica de espectro alargado para suporte conceptual e técnico do trabalho;

ii. Meios que têm como finalidade a quantificação e comparação das despesas que os organismos da APP tiveram com a aquisição/renovação de licenças de SP e com a contratualização de serviços associados ao desenvolvimento/implementação de tecnologias livres e de código aberto;

[30] Tradução livre do autor a partir do original, *"It is common to describe inductive inference as a "specific-to-general" reasoning process that uses a number of individual observations to generate laws that match these know observations and can be used to predict currently unknown future observations." In* CICHOSZ, Pawet (2015) – **Data Mining Algorithms Explained Using R**. John Wiley & Sons, 1ª Edição, Nova Iorque, 683 p.

iii. Métodos de recolha de dados sobre o estado da adopção de SL/CA na APP, nos quais se incluem experiências empíricas (de migração) e informações sobre dificuldades na tomada de decisão sobre migrar ou não;

iv. Técnicas específicas que visam a pré-selecção de *software* e a incrementação de um conjunto de procedimentos avançados de modelação espacial, tendo em vista a obtenção de dados sobre a consistência da sua performance.

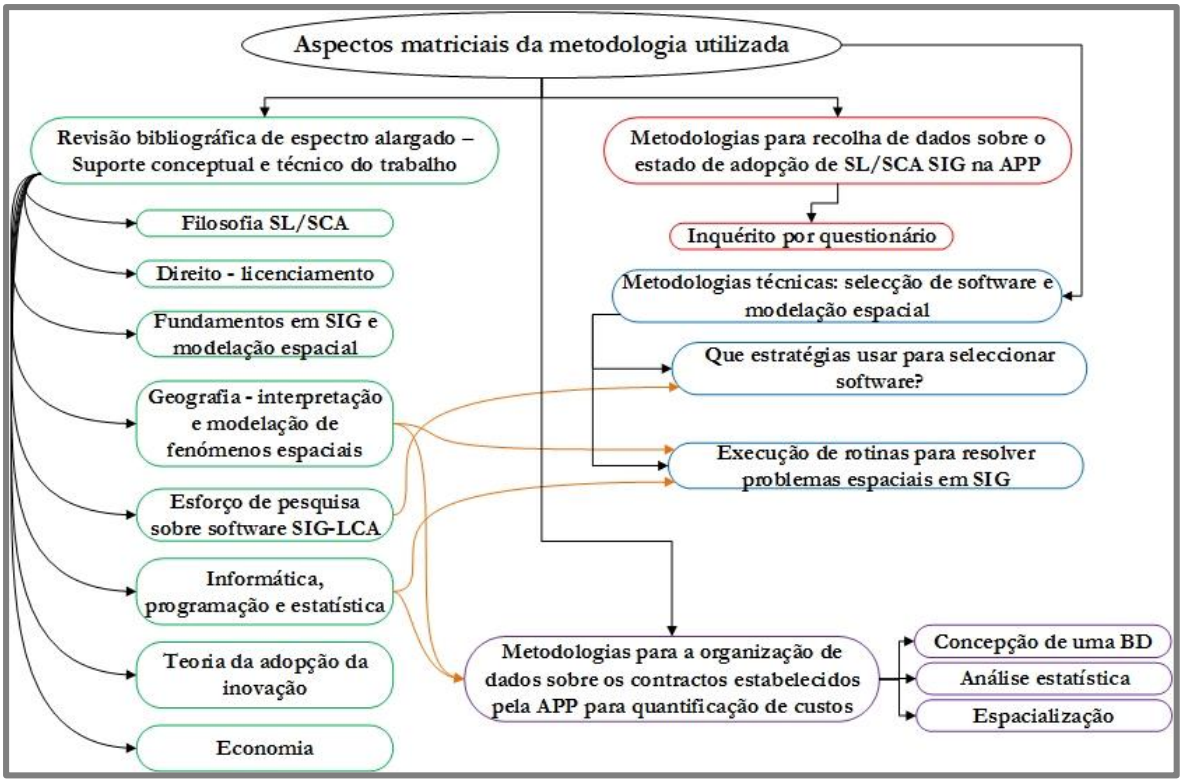


Figura 3 - Orgânica metodológica a aplicar no desenvolvimento da dissertação.

3.1. Revisão bibliográfica de espectro alargado

Parece pacífico dizer-se que o grau de maturidade de qualquer investigação depende da qualidade do seu suporte bibliográfico. O fenómeno do SL/SCA é complexo e gera inúmeras oportunidades de investigação que giram em torno de várias áreas do saber (Krogh & Spaeth, 2007). Na realidade, na esquematização deste projecto de investigação, e embora seja uma dissertação em TIG com enfoque na problemática da adopção de SL/CA SIG na APP, é impensável não alargar a revisão bibliográfica a uma panóplia de áreas do conhecimento, que, embora possam parecer, à primeira

vista, fora de contexto, estão directa ou indirectamente, relacionadas com os objectivos anteriormente definidos.

Assim, distinguem-se os seguintes campos científicos com interesse para o desenvolvimento desta exposição:

i. Fundamentos filosóficos que geraram e distinguem os conceitos de SL/CA e de SP, e que esclarecem os mitos associados aos primeiros – é fulcral conhecer o objecto de estudo.

ii. Direito (legislação): implicações legais das diferentes licenças que protegem cada tipo de *software* e a interpretação das questões relacionadas com a defesa dos direitos de autor; nível de consistência e integridade dos diplomas legais que regulamentam a implementação/avaliação de SL/CA no sector público.

iii. Consolidação da aprendizagem de métodos que nos podem auxiliar na escolha do *software* que iremos testar, e familiarização com os fundamentos do *software testing* e com a literatura que discute a construção de modelos que têm como propósito averiguar a qualidade global de um programa informático.

iv. Fundamentos, processos e aplicações (importância para o sector público) dos SIG (e modelação espacial), que, associadas à capacidade de interpretação dos fenómenos espaciais, servem de suporte (e contextualizam) à formulação de modelos (espaciais, lógicos e estatísticos) aplicados na bateria de testes que têm por objectivo avaliar o desempenho de diferentes programas SIG.

v. Conhecimentos em informática, computação e programação em Python – importantes na construção de rotinas/procedimentos a serem usados nos exercícios de desempenho.

vi. A psicologia das organizações e a teoria/modelação da adopção da inovação na organização são necessárias para sistematizar métodos de cálculo dos custos inerentes ao processo de migração.

vii. Por fim, são relevantes saberes na área da economia, fundamentais para a reflexão sobre conceitos como o de TCO e ROI.

3.2. Metodologia para organização e quantificação de dados relativos às despesas da Administração Pública Portuguesa com aquisição de *software*

Passemos então a debruçar-nos sobre os principais aspectos que suportam a obtenção e tratamento dos dados que possibilitam a contabilização dos custos com aquisição/renovação de licenças e aquisição de serviços relacionados com *software* (SP e SL/CA).

i. Elaboração de um inventário exaustivo dos dados armazenados na plataforma 'base: contratos públicos online' (http://www.base.gov.pt/Base/pt/Homepage) – Figura 4 – depois de identificados os critérios para a pesquisa, a saber:

Figura 4 - Plataforma 'base: contratos públicos online' (http://www.base.gov.pt/Base/pt/Homepage, acedido em 23 Janeiro, 2016).

a. Dados relativos a despesas das instituições da APP com aquisição/manutenção de licenças de *software* SIG proprietário, considerando-se 7 produtores/fornecedores;

b. Dados relativos a despesas das instituições da APP com a aquisição/manutenção de licenças de *software* proprietário relacionado com sistemas operativos (S4OS) – incluem-se também os próprios sistemas operativos -, considerando-se 5 produtores/fornecedores;

c. Dados relativos a despesas das instituições da APP com a contratualização de serviços relacionados com SL/CA (contratos estabelecidos pela APP em que a entidade adjudicatária fornece serviços suportados por soluções livres ou de código aberto), considerando-se 19 empresas fornecedoras deste tipo de serviço.

ii. Constituição de uma BD (Figura 5 e Tabela IV). Esta foi organizada de modo a sermos capazes de quantificar, comparar (pesar directamente os custos com *software* proprietário e serviços relacionados com SL/CA) e espacializar (cartografar por município) os dados relativos aos gastos totais para todos os itens acima referidos, considerando o período cronológico entre Agosto de 2008 e Outubro de 2015 numa tentativa de diagnosticar tendências, e ponderando o peso que cada um dos sectores da APP tem no total dos gastos[31]. Estes sectores identificam-se de acordo com as seguintes classes:

a. Educação;

b. Serviços Municipalizados (sector da administração local);

c. Administração Central e Regional;

d. Empresas de Gestão do Território e Recursos;

e. Saúde;

f. Segurança;

g. Outros serviços.

[31] Salientamos que este inventário diz apenas respeito a algumas das principais fornecedoras de *software* (SIG e S4OS), para o intervalo de tempo definido – os dados recolhidos foram considerados numa perspectiva de análise de evolução cronológica das despesas efectivas do Estado nesse período de tempo. A opção por esta cronologia não obedeceu a qualquer tipo de critério que não o da obtenção de dados confiáveis para um intervalo de tempo considerável, antecâmara da grave crise económica que viria a deflagrar um par de anos mais tarde, e da entrada em vigor do próprio OE para 2013, que, como se sabe, é profundamente limitativo quanto à disponibilização de subsídios para aquisição de SP havendo soluções alternativas em código aberto.

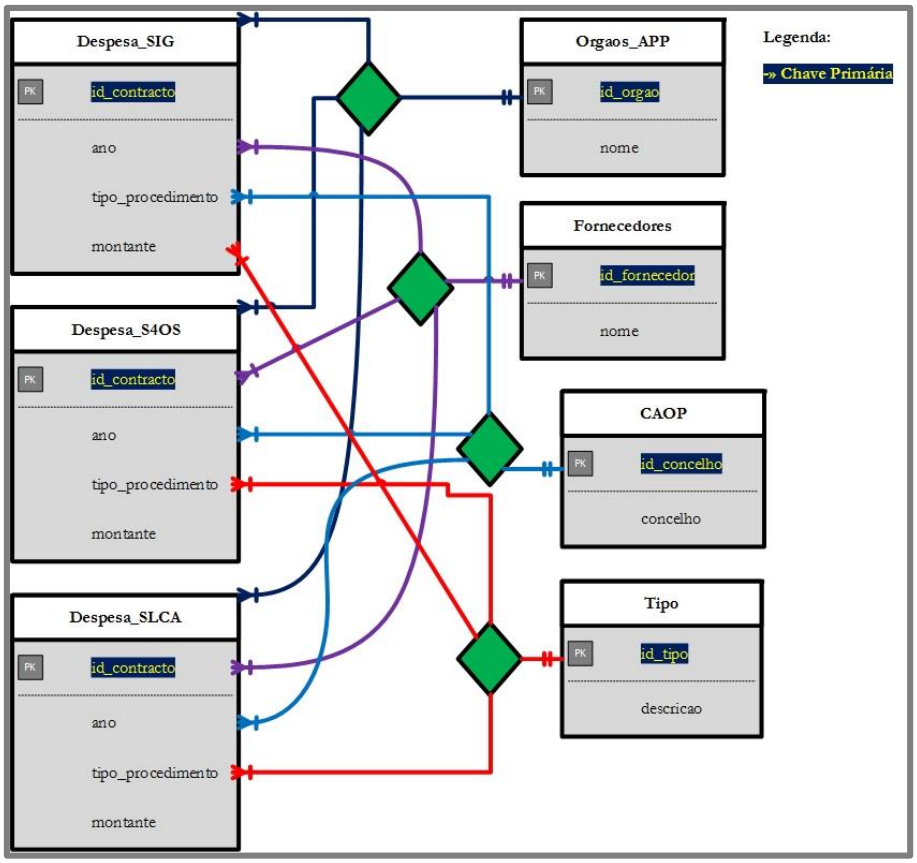


Figura 5 - Modelo Entidade-Associação (ME-A)[32] que estabelece a estrutura da BD construída para quantificação das despesas da APP com a aquisição/renovação de licenças e serviços relacionados com SP e SL/CA.

Tabela IV - Modelo Relacional (MR)[33] que estabelece a estrutura da Base de Dados construída para quantificação das despesas da APP com a aquisição/renovação de licenças e serviços relacionados com SP e SL/CA

Estrutura do Modelo relacional – 'relacao(chavePrimaria a negrito, campos, chaveExterna a sublinhado)
Despesa_SIG (**id_contrato**, ano, tipo_procedimento, montante, id_fornecedor, id_orgao, id_concelho, id_tipo) Despesa_S4OS (**id_contrato**, ano, tipo_procedimento, montante, id_fornecedor, id_orgao, id_concelho, id_tipo) Despesa_SLCA (**id_contrato**, ano, tipo_procedimento, montante, id_fornecedor, id_orgao, id_concelho, id_tipo)

[32] O Modelo Entidade-Associação (M E-A) representa o modelo conceptual da BD, sendo baseado na percepção de um mundo real que consiste numa colecção de objectos básicos, chamados entidades, e associações entre esses objectos, discriminando também um conjunto de restrições e chaves. Assim, o M E-A é uma linguagem humana que interpreta o funcionamento lógico da BD, aferindo a sua viabilidade, e preocupando-se com aspectos relacionados com a sua consistência, integridade e redundância. Com o conceito da BD num M E-A, a passagem para o Modelo Relaciona torna-se uma tarefa intuitiva e menos confusa.

[33] Este Modelo Relacional (MR) consuma a implementação do M E-A da Figura 5 (deixa-se a fase da conceptualização). No MR, as entidades transformam-se em relações (tabelas), os atributos materializam-se em campos e as cardinalidades são materializadas com recurso a chaves estrangeiras. Uma chave estrangeira (ou externa) é o campo de uma relação que corresponde à chave primária de outra relação, permitindo a associação entre as duas.

Orgaos_APP (**id_orgao**, nome) Fornecedores (**id_fornecedor**, nome) CAOP (**id_concelho**, concelho) Tipo (**id_tipo**, descricao)

Com esta BD torna-se possível a utilização de rotinas específicas que nos permitem reunir a informação já processada de modo a proceder às indispensáveis leituras e interpretações. Entre elas destacam-se os seguintes programas:

i. Programa 1[34] - cálculo de frequências absolutas e relativas de uma tabela, segundo intervalos previamente definidos pelo operador;

ii. Programa 2[35] - inquirição das relações que dispõem dos valores em numerário associado a cada contrato (*Despesa_SIG, Despesa_S4OS* e *Despesa_SLCA*), tendo em vista a criação de uma nova relação, cujas instâncias discriminam os gastos totais que cada organismo teve;

iii. Programa 3[36] - inquirição das relações que dispõem de informação sobre os contratos com o intuito de sistematizar os totais de despesa por empresa fornecedora/sector de actividade da APP e por ano;

iv. Programa 4[37] - bloco de código que estabelece a associação entre os contratos e despesas e a Carta Administrativa Oficial de Portugal (desagregação ao nível do concelho), possibilitando a espacialização dos valores totais de despesa por município da sede da instituição adjudicante.

Para complemento destas leituras, crê-se que a comparação directa entre as despesas com SP e SL/CA é importante como elemento que nos dá uma pista sobre qual é o mais oneroso. Porém, a obtenção desta diferença não é tão simples quanto possa parecer, visto que exige mais do que uma simples subtracção, isto porque o número de contratos que respeita ao SP é muito superior àquele que totaliza os contratos relacionados com SL/CA.

Sabendo que *N* (número total de contratos relativos a SP com montante associado) é maior que *M* (número total de contratos relativos a serviços SL/CA com montante associado), faz com que a subtracção da soma dos valores contratuais de cada elemento de *N* pela soma dos mesmos valores de cada elemento de *M* não devolva um valor lógico (estaríamos a comparar duas amostras com

[34] **Ver Anexo A, p. A1.**
[35] **Ver Anexo A, p. A2**.
[36] **Ver Anexo A, p. A2.**
[37] **Ver Anexo A, p. A3**.

um peso completamente diferente no mercado – o SP é muito mais usado e por isso tem muitos mais contratos associados do que o SL/CA).

No entanto, se *K* for uma amostra aleatória de *N* (sem elementos repetidos) com a dimensão de *M,* a subtracção dos somatórios dos valores contratuais referentes a cada elemento de *K* e *M* já faz mais sentido.

Ainda assim, como se imagina, se gerarmos duas amostras aleatórias, a subtracção dos seus somatórios com o somatório de *M,* provavelmente, devolveria resultados diferentes, ou seja, dependendo da amostra, o SP poderia implicar mais ou menos custos, situação que não satisfaz o propósito de comparação[38].

Com a finalidade de a tornar possível desenvolveu-se o Programa 5[39], que, admitindo *N*, gera, até cumprir *Z* (número de iterações definidas pelo utilizador), um número de *K* igual a *Z*, para, posteriormente, calcular $A_1, A_2, A_3, \dots, A_n$, valores que dizem respeito à soma dos elementos de cada amostra *K*. Assim, *O* é a média dos valores $A_1, A_2, A_3, \dots, A_n$, e representa o montante médio, em euros, para uma amostra *K* com a dimensão de *M*. Neste sentido, quanto maior for *Z,* maior será a diversidade das amostras, logo *O* chama praticamente todos os contratos à comparação com o numerário de *M*. Este algoritmo não ignora medidas de dispersão, nomeadamente o desvio-padrão ou a amplitude entre o total máximo e mínimo entre todas as amostras geradas.

3.3. Recolha de dados sobre o estado de adopção de *Software* Livre e/ou de Código Aberto, nomeadamente SIG, na Administração Pública Portuguesa

A obtenção de experiências comprovadas em matéria de adopção e superação (positiva ou negativa) de um processo de migração é uma tarefa árdua, no entanto, crucial para o desenvolvimento de uma aplicação como a SIGósAPp.

[38] Imagine-se que a nossa BD regista 1 000 contratos relativos à aquisição/renovação de licenças de SP e apenas 200 acordos que formalizam um pedido de implementação/suporte de uma tecnologia livre. Suponha-se que o montante total desses 1 000 é de 10 000 000 € e que o montante total dos outros 200 é apenas 3 000 000 €. Ao gerarmos uma amostra aleatória a partir dos 1 000 contratos de SP, o somatório de todos esses contratos poderia ser superior aos 3 000 000 € gastos com SL/CA, no entanto, se de seguida criássemos outra amostra aleatória, poderíamos ter uma situação em que o numerário total associado a esta segunda amostra fosse inferior aos 3 000 000 €. Enfim, ao compararmos amostras aleatórias com o mesmo número de elementos, a diferença da sua soma é proporcional à diversidade das amostras criadas.

[39] **Ver Anexo A, p. A4**.

É um requesto íngreme uma vez que os utilizadores não estão sensibilizados para estas questões, ou não têm conhecimento, ou, somente, não têm disponibilidade para partilhar a sua experiência ou opinião.

Antecipando a possibilidade de não virmos a obter resultados satisfatórios com a criação de um Fórum de discussão ou de um Grupo de Trabalho na plataforma *Facebook* (cujo interesse estaria na partilha de experiências e conhecimentos técnicos), e na impossibilidade de se realizarem entrevistas a actores de processos efectivos de migração, optou-se pela formulação de duas estratégias que visam a recolha de dados sobre o estado da adopção de SL/CA na APP.

Em primeiro lugar, achamos interessante completar quaisquer informações que viessem a ser adquiridas com a inferência de dados sobre o estado da adopção de SL/CA na APP a partir da BD que regista as despesas com aquisição/renovação de licenças de SP e contratualização de serviços relacionados com SL/CA, alargando a análise à escala nacional. Neste sentido, construíram-se pequenos procedimentos que identificam possíveis casos de implementação/migração de SL/CA:

i. Programa 6[40] - discrimina os tipos de despesa por município (despesa apenas com S4OS, apenas com SIG; com S4OS e SIG em simultâneo; apenas com a contratualização de serviços relacionados com SL/CA; e sem despesa).

ii. Programa 7[41] - quantifica o número de órgãos, por município, que estabeleceram, no período de tempo considerado, contratos com fornecedores de SP e com prestadores de serviços relacionados com SL/CA.

iii. Programa 8[42] - a partir dos contratos disponíveis, agrega numa nova relação as informações contratuais dos organismos que estabeleceram, no período de tempo considerado, contratos com fornecedores de SP e com prestadores de serviços relacionados com SL/CA.

iv. Programa 9[43] - tendo por base o resultado do Programa 8, identifica padrões, de modo a predizer qual a situação desse órgão em termos de migração.

Em segundo lugar, optou-se também pela formulação e divulgação[44] de um inquérito por questionário, que tem como público-alvo os funcionários dos vários organismos da APP, nomeadamente aqueles que trabalham com TIG.

[40] **Ver Anexo A, p. A4**.
[41] **Ver Anexo A, p. A5.**
[42] **Ver Anexo A, p. A6.**
[43] **Ver Anexo A, p. A7.**
[44] Este inquérito foi alojado na plataforma LimeSurvey (https://www.limesurvey.org/en/, acedido em 18 Novembro, 2015) – plataforma de código aberto -, a correr nos servidores ao cargo do Serviço de Gestão de Sistemas e Infra-Estruturas de Informação e Comunicação. Para a sua difusão, enviaram-se mensagens de correio electrónico para as

Apresentado em anexo (Anexo B), a divulgação do referido inquérito tinha como baliza geral a obtenção e recolha do maior número de informação possível sobre a adopção de SL/CA, considerando o maior número possível de tipos de órgãos da APP e abrangendo todo o território nacional (continente e ilhas), através da:

i. Inventariação de casos de sucesso;

ii. Recolha de dados sobre experiências de migração (bem sucedidas ou mal sucedidas);

iii. Organização de informações relativas a complicações sentidas no balanço entre os dois tipos de *software*;

iv. Obtenção de opiniões sobre o melhor modo de se compor um guia/manual de apoio à migração;

v. Catalogação de problemas (espaciais) reais, a serem usados nos testes de desempenho;

vi. Por fim, solicitação de envio de críticas e recomendações sobre os pressupostos (originais) deste trabalho.

Na sequência da sua divulgação, com o intuito de se proceder à análise dos resultados, projectou-se uma BD, de modo agilizar a obtenção de ilações substanciais para a consolidação da nossa meta através de variadas inquirições que nos devolvem respostas aos pontos elencados. A Figura 6 apresenta o ME-A que conceptualiza a BD estabelecendo as associações entre as várias entidades consideradas. Por seu turno, a Tabela V representa a transposição lógica para o MR do mesmo ME-A.

instituições que conseguimos inventariar em tempo útil, na expectativa de que elas dessem o seu contributo, respondendo e difundindo por outras.

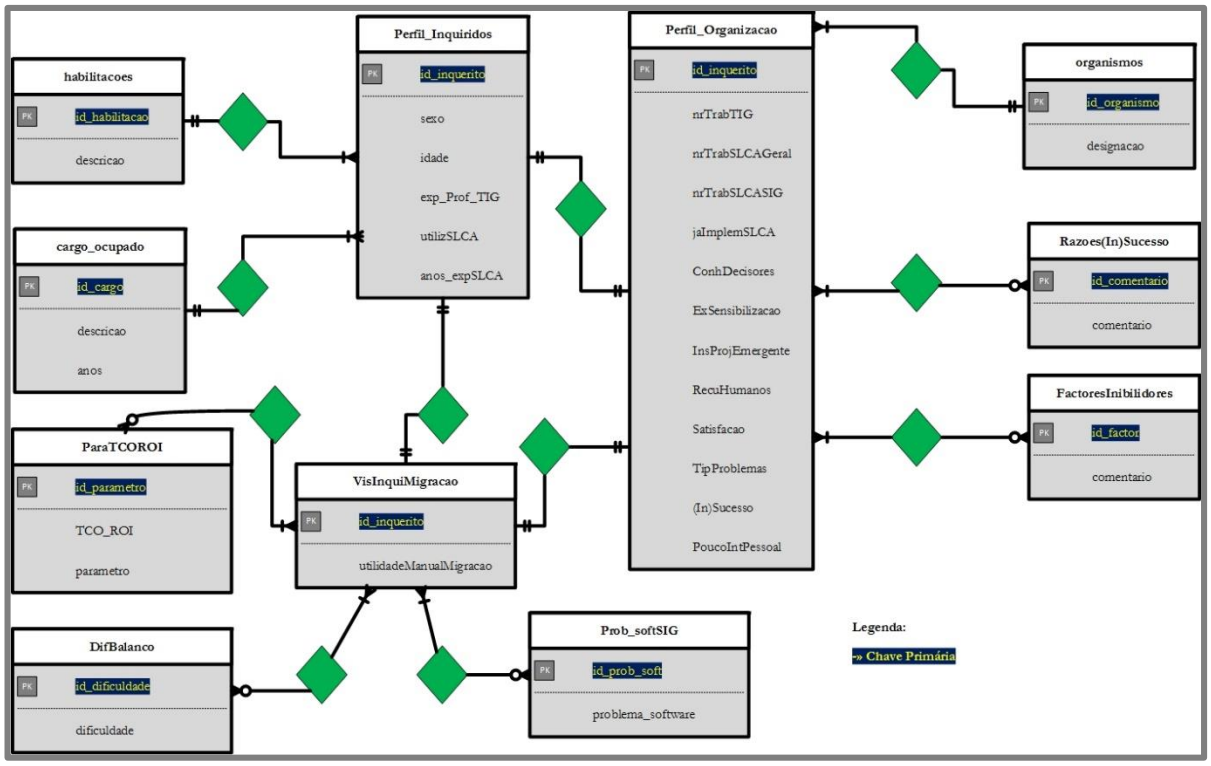


Figura 6 - Modelo Entidade-Associação que estabelece a estrutura da BD para análise das respostas obtidas com base no inquérito.

Tabela V - Modelo Relacional que estabelece a estrutura da BD para análise das respostas obtidas com base no inquérito

Estrutura do Modelo relacional – 'relacao(chavePrimaria a negrito, campos, chaveExterna a sublinhado)
Perfil_Inquiridos (**id_inquerito**, sexo, idade, exp_Prof_TIG, utilizSLCA, anos_expSLCA, id_habilitacao, id_cargo) habilitacoes (**id_habilitacao**, descricao) cargo_ocupado (**id_cargo**, descricao, anos) Perfil_Organizacao (**id_inquerito**, nrTrabTIG, nrTrabSLCAGeral, nrTrabSLCASIG, jaImplemSLCA, ConhDecisores, ExSensibilizacao, InsProjEmergente, RecuHumanos, Satisfacao, TipProblemas, (In)Sucesso, PoucoIntPessoal, id_organismo, id_comentario, id_factor) organismos (**id_organismo**, designacao) Rel_Razoes (**id_inquerito, id_comentario**) Razoes(In)Sucesso (**id_comentario**, comentario) Rel_Factores (**id_inquerito, id_factor)** nFactoresInibidores (**id_factor**, comentario) VisInquiMigracao (**id_inquerito,** utilidadeManualMigracao) Rel_TCOROI (**id_inquerito, id_parametro**) ParaTCOROI (**id_parametro**, TCO_ROI, parametro) Rel_DifBalanco (**id_inquerito, id_dificuldade**) DifBalanco (**id_dificuldade**, dificuldade) Rel_Prob_softSIG (**id_inquerito, id_prob_soft**) Prob_softSIG (**id_prob_soft**, problema_*software*)

As respostas aos questionários estão longe de serem as que necessitávamos, visto que a sua interpretação não nos permite retirar um número aceitável de conclusões relevantes.

Obtivemos uma amostra de 45 indivíduos num universo amostral potencial que ultrapassa os milhares, o que se traduz, segundo os autores que têm vindo a procurar fórmulas para cálculo do *sample size*, numa residual significância estatística. Porém, a nosso ver, a qualidade da amostra não se mede em termos do seu significado estatístico. Mais do que o elevado número de respostas, procurávamos esclarecimentos que valessem pela sua qualidade e não tanto pela sua quantidade. Pretendíamos, acima de tudo, que nos fossem apresentados problemas, soluções e sugestões.

Tínhamos como pressuposto inicial a análise dos resultados, não numa perspectiva de valorização dos números (embora a repetição de um argumento lhe atribuísse maior relevância), mas sim na interpretação lógica dos argumentos propostos, que nos deveriam indicar ou apontar um panorama geral de alguns órgãos, os quais se constituiriam como casos de sucesso ou insucesso, indicando-nos parâmetros válidos a incluir no nosso balanço de soluções informáticas.

Pelo contrário, e mais importante do que o reduzido número de inquirições obtidas, destaca-se o facto de serem raras as respostas com substância, resultados prováveis da falta de conhecimento e do desinteresse evidenciado pelos indivíduos inquiridos.

Em consequência, não foi possível reunir dados empíricos com o significado que se desejava, que nos permitam *a posteriori* consolidar um modelo de equação de custos ou um MDS4OSS. Reconhecemos que o inquérito é pesado e maçudo, todavia, não encontramos outra maneira de promover um escrutínio tão amplo quanto possível aos trabalhadores da APP, tendo em vista os objectivos que pretendíamos alcançar. Ainda assim, trata-se de uma amostra exemplificativa que não deixará de ser considerada no decurso do trabalho, na medida em que nos confirma algumas das nossas premissas iniciais e nos fornece conclusões interessantes, sobretudo no que diz respeito à falta de sensibilidade e consciencialização comum perante a importância destas questões, ignorando-se assim as imposições legislativas, o que, por sua vez, tem tradução em situações de potencial incumprimento da lei.

Por fim, com o intuito de tornar a análise dos inquéritos o mais célere possível, construiu-se o Programa 10[45], que concretiza um conjunto de métodos para identificação e quantificação das respostas dadas.

[45] **Ver Anexo A, p. A9.**

3.4. Breve nota sobre os avançados procedimentos técnicos para selecção das alternativas SIG a escrutinar e para avaliação da sua performance.

Entre as valências da multifacetada metodologia utilizada (cfr. Figura 3), definimos um grupo de métodos e procedimentos técnicos que, pela sua especificidade (enquadramento teórico/técnico e apresentação das decisões tomadas e da escolha das ferramentas) e por fazerem parte do cerne do trabalho, merecem uma explicação fundamentada e profunda em parte mais avançada deste documento, de modo a explicar ao leitor o que exactamente se fez.

Estes métodos têm como objectivo avaliar o desempenho (eficácia e eficiência) de diferentes programas SIG *Desktop*, adoptando uma perspectiva que nos conduz a reflectir também sobre problemas em torno da fundamentação falaciosa (?) de determinados mitos que referem que os resultados gerados por ferramentas ou processos análogos em diferentes programas não é o mesmo.

A Tabela VI elenca a bateria de testes que têm como finalidade o cumprimento destes objectivos. Antecedendo cada exercício, tendo em conta o elevado número de SL/CA SIG existente (Steiniger & Bocher, 2009; Chen *et al.*, 2010; Steiniger & Hunter, 2013), incrementou-se uma estratégia para circunscrever o universo de programas a testar, experimentando-se aquelas que em princípio têm condições para ser competitivas com o ArcGIS 10.2, segundo um conjunto de critérios previamente estabelecidos e discriminados. O ArcGIS 10.2 da ESRI foi o SP eleito para contraposição com o SL/CA SIG, por dois motivos essenciais: por um lado, é claramente o líder de mercado na área dos SIG, sendo o mais utilizado em maior parte dos organismos da APP que ainda usam SP; por outro lado, comparativamente a outras alternativas proprietárias (IDRISI, MapInfo, entre outros), o ArcGIS é o *software* que conhecemos melhor e é também o que se tem revelado mais polivalente, mais fiável e mais performante na maioria dos exercícios, sobretudo nos que envolvem tarefas mais complexas e/ou que envolvem maior volume de dados a processar.

Tabela VI - Bateria de exercícios realizados tendo em vista a avaliação comparada de desempenho entre diferentes soluções SIG *Desktop* (SP, SL/CA)

	Descrição	Legislação aplicável
1	Produção de cartografia de susceptibilidade a movimentos de vertente	Lei n.º 27/2006; Lei n.º 65/2007; Decreto-Lei n.º 166/2008
2	Modelação da acessibilidade geográfica dos indivíduos a infra-estruturas de saúde	Regulamento n.º 329/2013

3	Comparação estatística dos *outputs* gerados por um conjunto de ferramentas consideradas análogas	-

Voltando à Tabela VI, os dois primeiros exercícios tratam-se de problemas de modelação espacial em ambiente SIG consagrados na legislação portuguesa. O terceiro diz respeito a um conjunto de testes individuais, nos quais elencamos um conjunto de ferramentas transversais a vários *software* SIG *desktop*, que testamos tendo como fim a comparação estatística dos vários *outputs*, em busca de dissemelhanças entre eles, algo que, tratando-se de geoprocessos análogos, não devia acontecer.

Devemos dizer também que no desenvolvimento destes exercícios, privilegiou-se a concepção e utilização de procedimentos programados (em Python ou em *Model Builder*). Esta opção facilita a comparação dos tempos despendidos por cada *software* a concluir toda a cadeia lógica de geoprocessos, eliminando os períodos mortos de preparação das ferramentas (selecção dos parâmetros de entrada e de saída). Para além disso, com a programação deixamos de estar condicionados pela qualidade e disponibilidade de ferramentas (a inexistência de uma *tool* não condena à partida todo o processo), assim, superamos as restrições dos *software*.

A construção das várias rotinas depende do conhecimento que o operador tem sobre as propriedades e potencialidades do *software* com que trabalha, portanto, não é possível assegurar que as nossas rotinas tomam o caminho mais curto ou eficaz. No entanto, teve-se o cuidado de se construir programas com a mesma matriz lógica e que incrementam, sempre que possível, ferramentas análogas, possuindo um grau de similaridade que legítima qualquer comparação directa entre os vários resultados.

Como é óbvio, a realização destes exercícios não ignora os diferentes comportamentos que um mesmo *software* pode apresentar perante distintos volumes de dados (escalabilidade). Assim, no exercício 2 estão previstos vários sub-testes que têm como propósito avaliar a susceptibilidade da performance à utilização de *inputs* com mais memória, *features* ou células.

Temos também vindo a alertar para a necessidade de avaliação dos *software* não apenas no sentido do desempenho *tout court*, mas também ao nível da robustez e consistência científica da arquitectura das ferramentas análogas em confronto. Portanto, para confirmar se os vários *outputs* de procedimentos similares são idênticos, construíram-se dois programas em Python:

i. Programa 11[46] – calcula o coeficiente de correlação linear entre duas imagens matriciais (um coeficiente de 1 significa que a distribuição dos valores pela imagem é exactamente igual).

ii. Programa 12[47] – subtrai os valores dos pixéis, de modo a identificar e avaliar a dimensão de eventuais diferenças nos vários *outputs*.

Posto isto, importa também ter presente que qualquer ensaio de *benchmarking* será sempre subjectivo e incompleto porque depende de vários factores, desde logo, das competências e dos conhecimentos do operador, mas também do tipo de tarefas de análise e de geoprocessamento a serem ensaiadas, bem como das condições de ensaio. A este nível, salientamos que os testes realizados cuidaram de manter como elemento neutro todos os factores, conhecidos e parametrizáveis, que pudessem influenciar o desempenho do *hardware* (e. g. a configuração das máquinas em termos de velocidade de processamento e a minimização das tarefas simultâneas em curso – monitorizadas pelo gestor de tarefas), de maneira a garantir as mesmas condições de funcionamento da máquina aquando da realização das tarefas em cada um dos *software*.

Com o intuito de não falsear os resultados, os testes foram executados sempre na mesma máquina, uma com as seguintes características: processador Intel® Core™ i5-3330 CPU 3.00 GHz baseado em x64; 8 GB de memória RAM instalada; disco rígido com 465 GB; Windows 8.1 Pro de 64 bits.

Concluímos, explicitando a distinção entre ferramentas nativas e não-nativas, bem como a nossa posição relativamente à sua utilização. No presente trabalho importa-nos avaliar directamente soluções específicas e nativas, ou seja, na nossa óptica, se pretendermos contabilizar e comparar os tempos despendidos por diferentes *software* na execução de uma tarefa de interpolação como o *Inverse Distance Weighting* (IDW), não faria sentido, por exemplo, recorrer à ferramenta GDAL_GRID para avaliar o desempenho do QGIS, na medida em que esta constitui um dos geoprocessos da biblioteca GDAL/OGR, e o QGIS dispõe de uma *tool* própria para executar esta tarefa. O uso do GDAL_GRID conotaria de menor ou maior eficiência a biblioteca GDAL e não o QGIS. O mesmo vale para o emprego de ferramentas do GRASS GIS no QGIS. Enfim, esta nota tem por propósito fazer perceber o quão distorcida é a ideia que se transmite quando se diz que o *software* A concretiza uma tarefa com grande eficiência e eficácia, quando, na realidade, é o *software* B integrado no A que a viabiliza. Não obstante, não podemos deixar de fazer notar que esta lógica de integração e de interoperabilidade entre os vários módulos (repositórios e *software* SIG) em ambiente *Open Source* é uma valência que se estende a todo o universo do próprio *Open Source*,

[46] **Ver Anexo A, p. A10.**
[47] **Ver Anexo A, p. A11.**

facto que não pode ser deixado de lado no momento de considerar a opção de migração. Neste sentido, defendemos que, quando se fala na utilização de *software* SIG, não se deve colocar uma questão individualizada em jeito de "Qual é o teu SIG?" ou "Qual é o SIG com que trabalhas?"

1ª Parte –

Fundamentos da razoabilidade do trabalho e enquadramento terminológico, conceptual, normativo e legislativo do tema

CAPÍTULO I – SUBSÍDIOS EPISTEMOLÓGICOS PARA CLARIFICAÇÃO DOS CONCEITOS DE *SOFTWARE* LIVRE, *SOFTWARE* DE CÓDIGO ABERTO E *SOFTWARE* PROPRIETÁRIO

Qualquer investigação deve ter como premissa estruturante a formulação de novo conhecimento, imprescindível à resolução de um determinado problema. Este pressuposto implica o conhecimento prévio sobre o objecto ou realidade que se pretende estudar. Portanto, ao falarmos de SL/CA importa reflectir, antes de tudo, sobre a sua génese, e estabelecer pontos de contacto e de segmentação com o SP. Qualquer administrador público deve, em primeiro lugar, conhecer para definir o que pensa em adoptar, retirando desses conhecimentos as mais fiáveis ilações que lhe permitem fazer uma avaliação comparada entre alternativas informáticas correcta e consciente. Assim, este primeiro capítulo tem por principal missão a identificação e a reflexão crítica sobre alguns dos principais mitos associados ao SL/CA e combater a falta de conhecimento sobre este tipo de *software*. A sua estrutura pode definir-se da seguinte forma sumária:

1. ***Software* Livre e *Software* de Código Aberto; duas faces da mesma moeda**
   i. Comparar e reflectir sobre os conceitos de *Software* Livre e de *Software* de Código Aberto.
2. ***Software* Livre e/ou de Código Aberto – uma visão enegrecida ou crónicas de uma narrativa imprecisa e distorcida?**
   i. Desmitificar os principais mitos associados ao *Software* Livre e/ou de Código Aberto, pondo em evidência a sua verdadeira substância, combatendo a falta de conhecimento que se tem sobre ele.
   ii. Estabelecer um quadro que resuma as principais vantagens e desvantagens do *Software* Livre e/ou de Código Aberto.

1. *Software* Livre e *Software* de Código Aberto; duas faces da mesma moeda

> *"Se tu tiveres uma maçã e eu tiver uma maçã, trocando essas maçãs, continuaremos, cada um, a ter uma maçã. Mas se tu tiveres uma ideia e eu tiver uma ideia, trocando essas ideias, cada um de nós passará a ter duas ideias."*
>
> Georg Bernard Shaw[48]

O computador executa programas previamente armazenados na sua memória externa. Esses programas (*software*) definem-se como o código executável, ou instruções, que, ao serem transferidas para a memória interna e executadas pela máquina através da sua unidade de processamento central (CPU), controlam o comportamento e operações do computador, tendo em vista a obtenção de um resultado (Kennedy, 2009; Philips, 2009; Bouras *et al.*, 2013).

Nesta leitura simplista, reconhecem-se, portanto, no *software*, dois domínios fundamentais que se referem a propriedades intrínsecas ao seu modo de desenvolvimento (construção em ambiente fechado ou em comunidade), partilha, distribuição, prestação de suporte, melhoramento, e aos aspectos legais (licenciamento) e sociais associados (Figura 7), aspectos que nos recordam os modelos de "Catedral" e "Bazar" na perspectiva de Raymond (2001). Podemos assim, distinguir o processo de licenciamento em:

i. *Software* Proprietário (SP);

ii. *Software* que se caracteriza por admitir acesso ao código-fonte, podendo ser livre e de código aberto ou, então, apenas de código aberto (SCA)[49].

[48] No original, "*If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange the ideas, then each of us will have two ideas.*"

[49] Esclarecer desde já que, embora o SL e SCA sejam frequentemente designados pelo mesmo termo – *Software* Livre e de Código Aberto (SL/CA do inglês FOSS – *Free/Open Source Software*) – (Subramanyam & Xia, 2008; Blansit, 2009; Baytiyeh & Pfaffman, 2010; Perry & Margoni, 2010; Wang, 2012; Bouras *et al.*, 2013 ou Bouras *et al.*, 2014), objectivamente, eles não têm significado idêntico, como bem apontam autores como Wayner (2000), Stallman apud Gay (2002), Fuggeta (2003), Laurent (2004), Hentley & Kemp (2008), Lindberg (2008), Philips (2009), Steiniger & Bocher (2009), Steiniger & Hay (2009), Steiniger & Hunter (2013) ou Carillo (2014). Na sua incrementação prática e quotidiana não se denotam atributos que os distingam, tanto que muitos ignoram as suas diferenças e as perspectivas individuais de cada um dos movimentos que lhes deu origem (FSF e OSI).

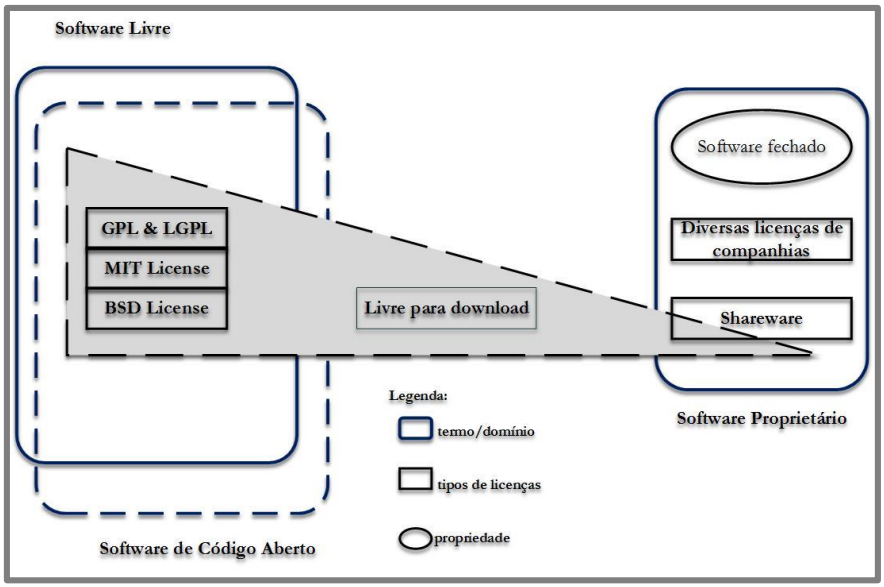


Figura 7 - Termos usados tendo em conta as licenças de *software*. Fonte: Steiniger & Bocher (2009), p. 3.

No confronto entre estes dois modelos é errado assumir-se que é a gratuitidade que os distingue, visto que ambos adoptam estratégias (diferentes, é certo) que visam a gestação de lucro. Consequentemente, é também erróneo referir que um é comercial e o outro não, do mesmo modo que não é verdade que um respeita os direitos de autor e o outro não. Ao contrário do que muitos querem fazer parecer, não existem só dissemelhanças entre eles. DiBona (2006) exalta que, em alguns casos, os desenvolvedores de SP são os mesmos que produzem SL/CA, por conseguinte, em alguns momentos, o SP e o SL/CA têm benefícios mútuos. Esta transversalidade tem o seu expoente máximo quando se emprega legalmente o conceito de *dual licensing*, segundo o qual um fornecedor disponibiliza o seu *software* sob duas licenças diferentes, uma que impõe regras típicas do SL/CA a certos utilizadores, e outra que disponibiliza para outros os termos clássicos do licenciamento proprietário (Olson, 2006).

Todo o *software* é gratuito, porém, é licenciado. Nesta constatação reside a verdadeira disparidade e profusão de conceitos e concepções difusas, posto que as licenças de ambos respeitam fundamentos legais distintos, nomeadamente no que concerne à protecção da propriedade intelectual, disponibilização e distribuição do código-fonte e restrição de usos.

Os *copyrights* (*all rights reserved*) ou direitos de autor, são automaticamente vinculados a uma nova expressão de uma ideia (Laurent, 2004; Olson, 2006; Lindberg, 2008), por conseguinte, o

desenvolvedor/fornecedor de um determinado programa tem a capacidade, segundo Philips (2009), de autorizar, ou não, os seguintes actos:[50]

i. Reproduzir o trabalho intelectual com direitos de autor associados em cópias;

ii. Preparar trabalhos derivados do trabalho original;

iii. Distribuir cópias do trabalho ao público por venda ou por qualquer outro tipo de transferência de propriedade (aluguer ou empréstimo, etc.);

iv. Executar o trabalho publicamente;

v. Revelar/mostrar o trabalho publicamente.

A lei dos direitos de autor aplicada à Propriedade Intelectual (Pereira, 2012) impõe o estabelecimento de um contrato entre o utilizador e o desenvolvedor/fornecedor de *software*, consumado pela aceitação, pelo primeiro, dos termos da licença (que consagram uma posição relativamente aos tópicos anteriores). O modo como estes parâmetros são regulamentados pela licença poderá implicar que um *software* se torne proprietário ou livre e de código aberto. Para que se torne mais claro o que pretendemos dizer com as observações precedentes, salienta-se que o facto de os parâmetros de uma licença vincularem o conceito de *copyleft* (que abordaremos nas seguintes linhas) não implica, segundo Laurent (2004), Lindberg (2008) e Philips (2009), que não existam direitos de autor ou de propriedade intelectual no sentido do conceito de *copyright*, pelo contrário, o primeiro implica desde logo que estes últimos estejam consagrados na licença.

Em moldes gerais, a licença do SP estipula a proibição da livre distribuição de cópias[51], e obriga o pagamento dos custos (compreendidos entre as centenas e os milhares de euros) associados à produção e distribuição das cópias, que, na prática, é uma compensação pelo respectivo *work of authorship* ou direitos de patente, no caso de o programa ser patenteado[52] (Laurent, 2004; Olson,

[50] Alguém que produza algo tido como uma obra intelectual tem, de modo inato e intrínseco, direitos sobre ela, ou seja, não é necessário qualquer processo burocrático que identifique alguém como o autor da sua produção intelectual, uma vez que os direitos ficam definidos automaticamente (Laurent, 2004; Lindberg, 2008). Resta ao autor capacitar-se de uma forma de comprovar que o trabalho é fruto do seu esforço independente, uma medida de segurança caso haja violações aos direitos. Não obstante, não se exclui a possibilidade de que, em casos específicos, seja necessário o registo da obra, nem que seja como prova de autoria.

[51] Não obstante, algumas empresas disponibilizam e distribuem gratuitamente (não de modo livre) algumas versões (*trial*) para uso temporário (período de experimentação). Vencido este espaço de tempo, é exigida a apresentação de um código de registo único (*serial key*) para o continuar a usar.

[52] As patentes são monopólios (limitados no tempo – máximo de 20 anos) que servem desígnios de protecção de invenções e desenvolvimentos tecnológicos. Ao desenvolver uma ideia, o inventor possui a habilidade de prevenir que alguém crie, use, venda ou importe a invenção (outro inventor, desenvolvendo a mesma ideia de forma independente sem conhecer aquele que já a patenteou, não poderá estabelecer nova patente sobre a invenção). Por sua vez, os direitos de autor (*copyright*) são limitações à expressão da ideia, logo não proíbe outras expressões da mesma ideia, por isso, não é tão forte como a protecção da patente (Laurent, 2004; Lindberg, 2008).

2006; Lindberg, 2008; Philips, 2009). Neste contexto, o *freeware* é aquele tipo de SP que não obriga ao pagamento desta compensação (o Google Earth é um exemplo).

Este tipo de licença é um contrato restritivo dos direitos do utilizador (restrições que servem para proteger o trabalho dos desenvolvedores), entre os quais estão a utilização e a cópia do *software* sem o cumprimento dos parâmetros legais estabelecidos na primeira (Olson, 2006). Exactamente para salvaguardar os direitos de patente (caso existam) e dos autores/produtores (*copyright*), o código-fonte deste tipo de *software* assenta no conceito de *black box*, é, portanto, fechado, secreto, de propriedade exclusiva, uma Catedral a que se pode aceder respeitando regras restritivas (Raymond, 2001), pelo que se torna inviável a melhoria e adaptação do código às necessidades.

Em contrapartida, o SL assenta numa filosofia suportada por quatro liberdades (Gay, 2002), formalizadas por Richard Stallman, aquando da criação do movimento livre e da FSF (Tabela VII).

Tabela VII - Princípios que caracterizam o SL e o SCA. Fonte: Gay (2002) e http://opensource.org/osd (acedido em 6 Abril, 2015)

***Software* Livre – 4 Liberdades**	***Software* de Código Aberto – 10 fundamentos**
1. *Executar o programa para qualquer propósito;* ***2.*** *Estudar o funcionamento de um programa (via código-fonte) e adaptá-lo às necessidades de cada um;* ***3.*** *Redistribuir a terceiros cópias das suas versões, possivelmente já modificadas (distribuídas de forma gratuita ou mediante pagamento);* [53] ***4.*** *Melhorar (desenvolver) o programa de modo a que as modificações se tornem públicas, para que a comunidade inteira beneficie da melhoria.*	***1.*** *Redistribuição livre - a licença não deve restringir a distribuição gratuita ou a venda de software enquanto componente de uma distribuição que contém programas de diferentes fontes;* ***2.*** *Disponibilização do código-fonte;* ***3.*** *Trabalhos derivados - a licença deve permitir modificações e produção de trabalhos derivados a partir de um software, alterações/actualizações que podem ser distribuídas sob os mesmos termos da licença original;*
4. *Preservação da integridade da autoria do código-fonte: a licença pode restringir o código-fonte de ser distribuído na sua forma modificada somente se a mesma permitir a distribuição de patch files juntamente com o código com o propósito de modificar o programa no momento da compilação; a licença pode exigir que os software derivados de um outro tenham um nome ou um número de versão distinto do original;* ***5.*** *Não discriminação contra pessoas ou grupos;* ***6.*** *A utilização do software não pode ser proibida em nenhuma área de actividade;* ***7.*** *Os direitos associados ao programa devem aplicar-se a todos aqueles que tiverem acesso a uma cópia, sem que haja necessidade de se obterem licenças adicionais;* ***8.*** *Os direitos associados a um programa não dependem do facto de ele estar inserido numa distribuição de software particular, isto é, se o primeiro for extraído do segundo, todos aqueles que usarem essa parte devem ter os mesmos direitos daqueles que usam a distribuição original como um todo;* ***9.*** *A licença pode colocar restrições a outro software que seja distribuído com o software licenciado;* ***10.*** *A licença deve ser tecnologicamente neutra – o software deve poder ser executado em qualquer plataforma ou interface.*	

[53] Enquanto as cópias de *software* proprietário são, quase sem excepção, cobradas, no SL/CA, embora se considere razoável o pagamento de uma taxa, as cópias raramente implicam custos, e, se envolverem, são meramente simbólicos (regra geral).

Estas liberdades, ao contrário do SP, definem que o acesso ao código-fonte é um pré-requisito que as garante e sustentam todo o processo de desenvolvimento do pensamento.

1.1. Um pouco de história

O SL remonta aos inícios da década de 70, ainda que não tivesse esta designação. Enquanto funcionário do Laboratório de Inteligência Artificial do MIT, Richard Stallman fez parte de uma comunidade (*hackers*) de partilha de *software* (Gay, 2002). Em Janeiro de 1984, com o colapso dessa comunidade (que se deu nos inícios da década de 80), Richard começou a escrever o primeiro *software* GNU (*Gnu is not Unix*), dando, assim, origem à FSF, criada em 1985 (Williams, 2010). Como consequência, surgiu também uma licença do tipo *copyleft*, a GPL (*GNU General Public License*, por vezes, referida por GNU-GPL), escrita em 1989 pelo próprio fundador do movimento "Livre" (Henley & Kemp, 2008). Stallman defendia que lançar um programa para o domínio público sem quaisquer direitos de autor é o método mais eficaz de o tornar livre, no entanto, esta forma de proceder admite que qualquer um está autorizado a transformar o SL em SP. Assim nasce o *copyleft*, cujo objectivo é garantir a liberdade de utilização e mantê-la em projectos ou versões derivadas de *software* originalmente livre (Gay, 2002; Fuggetta, 2003; Laurent, 2004; Lindberg, 2008; Philips, 2009; Steiniger & Bocher, 2009; Williams, 2010; Steiniger & Hunter, 2013).

A GPL, por ser *copyleft*, não só concebe as quatro liberdades como as protege[54], tendo necessariamente presente o *copyright* pelos motivos que se explicaram anteriormente. Ora, tal significa que o único *software* realmente livre é aquele que se distribui mediante a licença do tipo GPL, visto que é aquela que garante a liberdade, não só dos utilizadores actuais, como a dos futuros, ao contrário de outras que não têm obrigações deste tipo (Tabela VIII).

Tabela VIII - Principais características dos diferentes tipos de licenças de SL/CA. Adaptado de Laurent (2004), Lindberg (2008), Philips (2009) e Godinho (2012)

Permissões	Licenças
Distribuição permitida	GPL; LGPL; *Public Domain*; BSD; MIT; Apache; *Artistic*; MPL; AFL

[54] A GPL é, certamente, a licença que incorpora o espírito do *copyleft* ao extremo. Outras licenças desenvolvidas no âmbito do movimento livre mostram-se menos exigentes, serve de exemplo a GNU Lesser/Library General Public License (LGPL). A LGPL, ao contrário da GPL, decreta que os termos da licença apenas podem ser impostos ao programa difundido com essa licença e não a outros programas que possam ser conectados a ele ou que aproveitem as suas funcionalidades (Steiniger & Hay, 2009). Por exemplo, se uma biblioteca que contém ferramentas de análise espacial for coberta por GPL, ela não pode ser usada pelo ArcGIS (ESRI), uma vez que este tem licença proprietária - a GPL implicaria que os seus termos se estendessem, neste caso, ao ArcGIS.

Sem restrições à utilização	GPL; LGPL; *Public Domain*; BSD; MIT; Apache; *Artistic*; MPL; AFL
Código-fonte disponível	GPL; LGPL; *Public Domain*; BSD; MIT; Apache; *Artistic*; MPL; AFL
Alteração do código-fonte	GPL; LGPL; *Public Domain*; BSD; MIT; Apache; *Artistic*; MPL; AFL
Modificações permanecem livres	GPL; LGPL; MPL
A ligação com o *software* proprietário é permitida	LGPL; *Public Domain*; BSD; *Artistic*; MPL; AFL

Ao contrário da GPL, licenças como a BSD (*Berkeley Software Distribution*), a MPL (*Mozilla Public License*), a MITL (*Massachusetts Institute of Technology License*), a AFL (*Academic Free License*) ou a *Apache License* integram-se nos preceitos do movimento da OSI, criado em 1998 por Bruce Perens e por Eric Raymond, que introduziram uma expressão de leitura mais "objectiva e mais técnica" – "código aberto", com o intuito de manter alguma distância relativamente aos princípios intrínsecos do SL, de modo a atrair a atenção de um número mais alargado de corporações de *software* (Carillo, 2014).

O movimento *open source* não ignora as liberdades básicas do SL e fomenta as mais-valias advindas do acesso ao código-fonte, assumindo, porém, uma maior flexibilidade em termos de licenciamento, visto que um SCA não tem estritamente de se reger pelo princípio do *copyleft*, de acordo com as dez orientações (concebidas pela OSI) que a licença de um SCA terá de cumprir para ser considerado como tal (cfr. Tabela VII).

A contraposição das quatro liberdades (de Stallman) com os dez pontos que definem o SCA revela que ambos representam uma orientação filosófica e postura assumidamente individualizada, de acordo com o esquema da Figura 7.

Como bem aponta Steiniger & Bocher (2009), na linha de argumentação de Gay (2002), o termo 'código aberto' apresenta insuficiências, pois apenas nos diz que o código está acessível a qualquer utilizador, podendo, por isso, ser estudado, no entanto, a modificação e distribuição, a partir do momento em que os desenvolvedores assim pretendam, estão sujeitas a condicionalismos e limitações.

Este ponto (cfr. Figura 7) individualiza os dois movimentos e traduz-se, como se disse, numa maior flexibilidade do SCA (ao nível da decisão da sua evolução paralela com o conceito de SL), na medida em que, na distribuição posterior de projectos modificados, é deixado ao critério dos programadores a possibilidade de manter o *software* com o código aberto ou de o tornar proprietário.

A terceira orientação estabelecida pela OSI possibilita isso mesmo, nas palavras disponíveis em http://opensource.org/osd (acedido em 6 de Abril, 2015), *the license must allow modifications and direved works, and must allow them to be distributed under the same terms*. Repare-se na utilização da expressão *must allow* - 'deve permitir' -, esta conjugação semântica, na nossa interpretação, não implica obrigatoriedade, mas apenas a "possibilidade de…", caso os desenvolvedores estejam interessados em manter o seu programa aberto.

Como refere Stallman, citado em Gay (2002), esta "liberdade de decisão" confere ao SCA um estatuto de eleição preferencial, em detrimento do SL, concepção que surge bem visível aos olhos de quem, como nós, procede a uma exaustiva revisão da literatura da especialidade – grande parte dos autores ignora ou desconsidera a expressão "*Software* Livre", sendo bem evidente a preferência pela utilização de "*Software* de Código Aberto", ainda que esta carregue, por vezes, um significado mais lato que integra também o primeiro conceito (Lerner & Tirole, 2001; Kovács *et al.*, 2004; Glynn *et al.*, 2005; Hwang, 2005; Bitzer & Schröder, 2007; Krogh & Spaeth, 2007; Federspiel & Brincker, 2010; Hauge *et al*, 2010; Oram, 2011; Qu *et al.*, 2011; Ghapanchi & Aurum, 2012; Li *et al.*, 2012; Marsan *et al.* (2012a); Marsan *et al.* (2012b); Spinellis & Giannikas, 2012; Engelhardt & Freytag, 2013; Marsan & Paré, 2013; Papadopoulos *et al.*, 2013; Singh & Holt, 2013; Kuwata *et al.*, 2014; Reisinger *et al.*, 2014; Sarrab & Rehman, 2014; Gallego *et al.*, 2015).

A leitura da bibliografia e os dados recolhidos no âmbito desta investigação (inquérito) sugerem que os utilizadores interessam-se por questões aplicadas e técnicas (fiabilidade, segurança, estabilidade, etc.), deixando à margem filosofias de promoção da liberdade (Ven & Verelst, 2010), o que traduz, necessariamente, numa leitura que permite perceber uma certa desvalorização do SL, que só se compreende se pensarmos como Fuggetta (2003), quando se refere às (polémicas) acusações que o conotam de algum fundamentalismo ideológico.

De facto, os princípios da FSF assumem uma dimensão notoriamente filosófica, apoiada no direito de liberdade e na ideia de que a livre partilha de conhecimento trará valor acrescentado à sociedade; ao invés, o movimento *open source* coloca em evidência motivações técnicas/económicas, e os benefícios que os vários utilizadores terão com a possibilidade de acederem ao código-fonte (Seltzer, 2006), pelo que não é estranho dizer-se que este movimento é, sobretudo, um modelo de negócio, ou que pelo menos tem evoluído nesse sentido, atribuindo-se-lhe o nome de *open source 2.0*, tal como sugerem Mahony & Naughton (2004), Fitzgerald (2006), Li *et al.* (2012) ou Reisinger *et al.* (2014). Esta perspectiva desmonta a metáfora do modelo de 'Bazar' proposta inicialmente por Raymond (2001). Em síntese, embora as inconformidades conceptuais entre ambos os conceitos sejam plausíveis mas pouco visíveis do ponto de vista teórico – os dois movimentos chegam a

fundir-se no conceito de FLOSS (*Free/Libre Open Source Software*), elas existem na prática, e merecem destaque pelas implicações que podem ter[55].

Em jeito de conclusão, entendemos que independentemente de qualquer apego individual às orientações destes movimentos, as três abordagens de desenvolvimento de *software* (livre, código aberto e proprietário) são consideradas válidas, tendo todas elas benefícios e desvantagens que importa ponderar. Na sua substância, os paradigmas em causa têm uma posição no mercado que tem como objectivo a dinamização da economia com a satisfação das necessidades dos utilizadores, potenciando actividades de gestão da sociedade e iniciativas de investigação ou empresariais:

i. O SP cobra o esforço e energias do trabalho tido na sua produção, factor contemplado na licença que legaliza os direitos de patente e de autor, providenciando um conjunto de serviços de suporte, caso estes sejam contratualizados;

ii. Os movimentos livre e de código aberto distribuem o *software*, por norma, gratuitamente (embora possa não ser assim – como anteriormente se tentou demonstrar) e garantem o acesso ao código-fonte (o que permite modificações e adaptações aliciantes), sendo um modelo mais vocacionado para a prestação de serviços relacionados, por exemplo, com o suporte a um determinado sistema de TI ou com a formação do pessoal de um determinado organismo ou empresa.

Resta saber qual é a opção que, comprovadamente, é mais vantajosa para um dos diversos contextos da APP, pois não há soluções sem custos. Embora o SL/CA seja uma alternativa de baixo custo, dada a sua complexidade levantam-se outros factores que podem implicar custos potenciais, os quais devem calibrar a equação para cálculo do seu preço total no momento da tomada de decisão por uma ou por outra das soluções (SL ou SP).

1.2. *Software* SIG Livre e/ou de Código Aberto – homogeneidade na variedade

[55] A democratização do programa, para alguns desenvolvedores, pode, hipoteticamente, dar fôlego a uma difusão que fomentará o uso do seu *software*, estimulando migrações. Assumida a opção pela migração, depois de os vários organismos terem as suas soluções suportadas pelo SCA (ou pacotes de SCA eleitos, que, por definição, são integralmente interoperáveis por obedecerem às normas e especificações abertas, note-se!), no caso de poder vir a ser alterado o modo de licenciamento numa versão actualizada (com melhorias críticas) para um modelo proprietário, na impossibilidade de novo processo de migração, os decisores poderão ser sempre confrontados com a imposição de custos que, inicialmente não estavam previstos. Tudo isto concorre para uma certa confusão e complexidade no campo das regras de um jogo ainda muito imaturo, facto que concorre para a manutenção de alguma inércia perante a possibilidade de se optar pela migração para SL/CA.

Os SIG integram-se no amplo universo das TIG (Figura 8). O termo TIG emprega-se para designar o conjunto de instrumentos de gestão de informação geográfica, e assumem-se hoje como ferramentas centrais para a gestão do território e de todas as actividades que envolvam uma componente espacial[56] (Cosme, 2012).

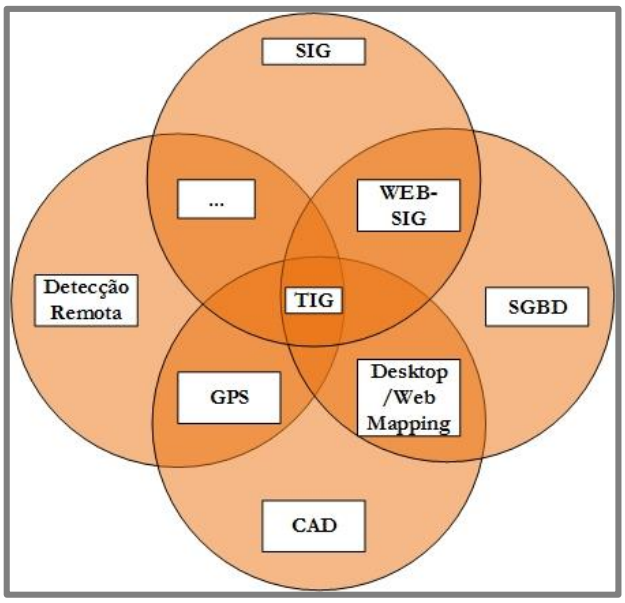


Figura 8 - As TIG e suas principais ramificações. Adaptado de Maguire (1991), p. 13; Konecny (2003), p. 11 e Cosme (2012), p. 6.

A aplicabilidade dos SIG e o seu serviço para o bom desenvolvimento das sociedades são inegáveis, isto porque só é viável gerir o que se conhece – são as informações espaciais, inferidas de dados reais, que permitem a formulação de conhecimento conducente a uma boa tomada de decisão em matéria de ordenamento do território e gestão ambiental.

Mas, como é que se pode definir o conceito de SIG? A dificuldade em encontrar uma definição com aceitação universal para o conceito de SIG surge, desde logo, com os múltiplos significados que a letra 'S' pode encerrar (Burrough & McDonnell, 2000; Longley *et al.*, 2005; Goodchild, 2009a e 2010):

i. Sistema - sistema de informação que contém informação georreferenciada (localização relativa a um referencial) estruturada e organizada, e é composto por um conjunto de

[56] Praticamente tudo o que acontece, acontece em algum lugar (Longley *et al.*, 2005; Matos, 2008). Nos seus recantos euclidianos, a superfície terrestre é infinitamente complexa, quer olhemos para ela como um objecto físico com características singulares em cada ponto da sua diversidade, ou a vejamos como o suporte da vida e desenvolvimento das territorialidades humanas. Esta complexidade suscita problemas espaciais tão diversos que a sua solução passa inexoravelmente pelo emprego das TIG.

elementos e processos interconectados como um todo que analisam e processam a informação segundo uma sequência de geoprocessos, que culmina com a obtenção de um ou vários *outputs*;

ii. Ciência (do inglês *Science*) – por um lado, os geoprocessos incrementados e incluídos nos pacotes de *software* são fruto da ciência, da matemática e da computação; por outro lado, a lida com um SIG é um procedimento que implica que o operador empregue premissas científicas intrínsecas ao problema que tenciona solucionar, concedendo maturação lógica e científica ao processo executado;

iii. Sociedade - dimensão que transcende a ciência geográfica e afasta os SIG da mera plataforma informática. Com o desenvolvimento e democratização, em específico, dos SIG, e das TIG, os cidadãos, enquanto neogeógrafos (Goodchild, 2009b; Graham, 2010), participam activamente como produtores e utilizadores (iletrados) de informação geográfica através de diferentes geoportais (Goodchild, 2009a) disponibilizados para o efeito e de outras plataformas[57]. Este desenvolvimento promove novas formas de experienciar o espaço geográfico, fomentando multiterritorialidades (vivências, acções, sentimentos) em territórios virtuais (representação virtual).[58]

Neste contexto, subsistem múltiplas interpretações do significado de SIG, cuja inventariação seria árdua, morosa e despropositada. Detalhamos apenas os parâmetros que, na nossa interpretação, sustentam o conceito e lhe conferem propriedades de um sistema integrado (Figura 9), reinterpretando o que já foi defendido por outros autores como Maguire (1991), Openshaw (1991), Burrough & McDonnell (2000), Julião (2001), Longley *et al.* (2005) ou Cosme (2012):

i. Antes de mais, um SIG carece sempre de um propósito, um objectivo, um problema para resolver. Ao trabalhar com SIG, estamos sempre a resolver problemas relativos ao espaço e aos sistemas e agentes que nele interagem.

[57] A Neogeografia deve ser entendida como uma deturpação dos papéis tradicionais de produtor, transmissor e consumidor de informação geográfica (transformação da Geografia académica numa geografia do senso comum, que todos dominam). Apesar da aparente desvalorização da Geografia, a sua relação com a Neogeografia é simples. Não há qualquer sobreposição entre os dois termos, são coisas que ocorrem em paralelo, que não se controlam ou, no mínimo, são de difícil controlo, uma vez que, as pessoas, havendo meios para isso, são livres de manipular os dados e informação geográfica, alheados de qualquer rigor e qualidade (p. e., exactidão posicional e temática da informação disponibilizada).

[58] Inclusive, estas representações virtuais do espaço geográfico, por serem desiguais (normalmente, territórios de baixas densidades são menos representados), têm consequências a vários níveis - social, económico, cultural, etc., bem como na mutabilidade da paisagem, ou seja, maior densidade de objectos digitalizados fazem de um local uma rugosidade que, ao sobressair no espaço euclidiano (palimpsesto), atrai inúmeros tipos de fluxos, que, inexoravelmente, trazem alterações para a paisagem e modos de vida (Graham, 2010).

ii. Em segundo lugar, a concretização do objectivo depende de um conjunto de princípios e formulações matemáticas que tratam o espaço como algo parametrizável, um objecto com propriedades que, ao serem derivadas de uma abstracção e simplificação da realidade em camadas (*layers*), têm apetência para serem agregadas num algoritmo que se destina à resolução do problema – capacidade intelectual de trabalhar correctamente dados espaciais ou geoprocessamento.[59]

No âmbito da modelação espacial (abstracção e simplificação da realidade em temas georreferenciados, combinados de modo lógico com o intuito de produzir uma solução possível para responder ao problema de raiz), esta concepção faz com que um SIG possa ser entendido enquanto conceito independente do *software*, na medida em que aquilo que importa é o conceito, o procedimento (cadeia lógica de raciocínios que nos conduzem à solução) ou o conjunto de teorias matemáticas, métricas espaciais e lógicas, usadas, de modo sequenciado, com o fim de se obter um *output* final para o apoio à decisão em matéria de ordenamento do território[60]. Inclusive, é frequente ler-se que o SIG surge antes da informática – Konecny (2003), por exemplo, distingue o SIG clássico (analógico) do SIG moderno (computação). É, por essa via, legítimo pensarmos em diversas expressões que são frequentemente utilizadas na literatura da especialidade, tais como, "… em ambiente SIG" e "raciocínio SIG", que revelam um enfoque na componente imaterial e, não tanto, na lógica processual, operacional e técnica associada ao *software* e ao hardware utilizados.

A computação funciona apenas como um agente que garante eficiência e ânimo ao esforço de manipulação de dados espaciais. Assim, o SIG, a nível da execução prática dos procedimentos, seria considerado como refém do programa informático, de tal modo que aparece sempre associado à máquina e a um conjunto de ferramentas com enunciado numa qualquer linguagem de programação. É por motivos de incrementação eficiente dos conceitos inerentes à topologia e álgebra de mapas, que o *software* e, por conseguinte, o hardware, constituem partes estruturantes da anatomia de um SIG, segundo a maioria dos autores que problematizam a sua definição (Maguire, 1991; Longley *et al.*, 2005; Cosme, 2012) [61].

[59] Estratégias para recolha, armazenamento, pesquisa, análise, representação, visualização e publicação de dados geográficos.

[60] Teoricamente, uma análise de proximidade pode ser efectuada sobre um pedaço de papel, uma rede irregular de triângulos pode ser construída analogicamente, ou seja, cada uma das ferramentas de um determinado *software* SIG tem pressupostos matemáticas que, uma vez conhecidos e compreendidos, fazem o sistema funcionar.

[61] Assim, um SIG distingue-se daquilo a que comummente se designa por *desktop mapping*, visto que o segundo se resume a um conjunto de interfaces e mecanismos que favorecem a facilidade de utilização, pesquisa espacial, representação gráfica e produção de cartogramas temáticos (não permite geoprocessamento). Ao invés, para os SIG, a produção de mapas é, frequentemente, um passo intermédio no processo de análise ou um mero *output* disponível de uma complexa análise espacial.

iii. Claro que, tanto o propósito inicial como o algoritmo (sequência lógica de pressupostos matemáticos que conduzem à resolução do primeiro), perdem a sua essência sem um conjunto de dados georreferenciáveis no espaço euclidiano. Sem *inputs* não há *outputs*. E, *inputs* de má qualidade, incoerentes com a realidade, geram *outputs* desajustados a uma decisão ou premissa que se pretende formular. Sem dados, sem a sua correcção, sem o seu ajuste, sem a sua organização numa base de dados[62] (concebida em harmonia com a complexidade de relações entre a informação), o SIG não tem materialização!

iv. Por último, um SIG necessita de um operador, alguém não só capaz de manipular dados geográficos com recurso a um computador e *software*, mas dotado de capacidades científicas na área do conhecimento onde o problema em questão se desenlaça[63].

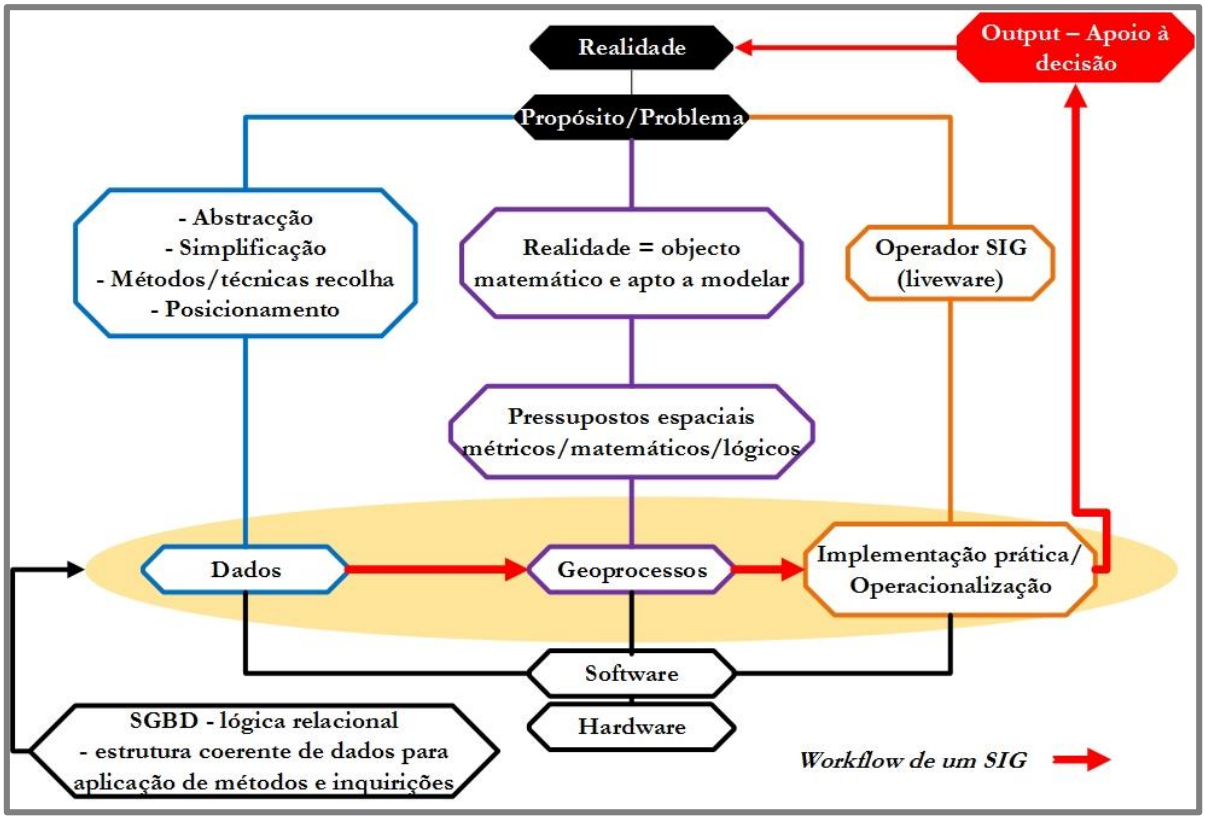


Figura 9 - SIG enquanto sistema integrado de processamento de dados espaciais composto por três dimensões (onde o *software* é apenas uma plataforma onde o processo corre), e embebido pelo propósito de apoiar a solução dos problemas da realidade. Adaptação do autor, com base

[62] Isto faz com que o conceito de SGBD (Sistema de Gestão de Base de Dados) seja associado ao conceito de SIG.

[63] O engenheiro informático produz *software* SIG, contudo, por não ter uma formação base em geografia, geologia, arqueologia ou engenharia geográfica, terá certamente sérias dificuldades em resolver, do modo mais adequado, determinado problema espacial. Da mesma maneira, no seio da Neogeografia, a utilização de plataformas (IDE), como o *Open Street Maps* (OSM) – https://www.openstreetmap.org/ -, que permitem, a qualquer um sem conhecimentos em Geografia, Engenharia Geográfica ou SIG, criar e editar dados espaciais -, não o requalifica como operador de SIG (Goodchild, 2009b; Graham, 2010).

nas obras de Maguire (1991), Openshaw (1991), Burrough & McDonell (2000), Julião (2001), Longley *et al.* (2005) e Cosme (2012).

Tudo isto para vincar uma ideia muito simples – o raciocínio é independente do *software*. Com o crescente desenvolvimento da capacidade dos computadores processarem dados e com o avanço do SL/CA, são cada vez mais os *software* com capacidades de manipulação, análise e processamento de informação geográfica. Esta diversidade pode assustar aqueles que iniciam a sua aprendizagem na área dos SIG, ainda que, em nosso entendimento se trate de uma concepção apriorística absolutamente despropositada e injustificada. Quando se fala em adopção de SL/CA SIG é legítimo avaliarem-se custos, eficiências e performances, exactidões lógicas e matemáticas dos *outputs*, bem como outros parâmetros de qualidade, na medida em que, todos eles são argumentos que fundamentam a tomada de decisão no momento da escolha de um *software*. No entanto, a necessidade de aprendizagem de um novo *software* não pode servir como justificação para a não implementação de SL/CA, designadamente por se tratar de *software* SIG – recordamos, com propósito o que antes se referiu sobre o "raciocínio SIG".

Quando nos iniciamos na utilização de uma nova interface, certamente que terá de haver um período de adaptação, uma curva de aprendizagem, de modo a nos familiarizarmos com alguns pormenores específicos da interface gráfica do programa, localização e organização taxonómica dos algoritmos. Em alguns casos em que se procuram migrações mais céleres, a frequência de acções de formação pode ser a solução, e existem muitas fontes de acesso gratuito a essa formação, quando estamos no domínio do SL. Exemplo disso não os *webinares*, uma modalidade de *e-learning* que democratiza o acesso a este tipo de formação técnica. Por outro lado, o processo de adaptação à nova interface é facilitado pelo facto de todo o *software* SIG *Desktop* possuir um ambiente gráfico que obedece a uma estrutura com uma matriz comum (Figura 10), reunindo um mesmo conjunto de propriedades e elementos, comandos, botões, etc., como, por exemplo:

i. Vista ou *view*, onde os pontos (individuais ou organizados em vectores) ou células dos vários temas são representados sobre um eixo cartesiano bidimensional, cujo ponto central diz respeito:
    a. à intersecção entre o semi-meridiano de referência (*Greenwich*) e o círculo máximo do Equador, no caso de se usar um sistema de referência com coordenadas geográficas;
    b. ao ponto central de projecção, no caso de se usar um sistema de referência com coordenadas planas, rectangulares ou projectadas;

c. ou a uma falsa origem, numa situação em que o referencial de projecção prevê uma deslocação do ponto central para um ponto fictício com o objectivo de colocar uma área específica no 1º quadrante[64].

ii. Tabela de conteúdos ou *table of contents* que permite ordenar os vários temas incluídos no projecto segundo o conceito de sobreposição de camadas ou *overlay* o que implica uma organização correcta e intuitiva dos temas (*layers*), por exemplo, a Camada A, quando posicionada em primeiro na tabela, que a Camada B, sobrepõe-se a esta; como consequência, na área de visualização – vista, se, por exemplo, a primeira for um tema pontual e a segunda uma imagem matricial, as duas são visíveis; invertendo a sua ordem na tabela, a primeira deixará de ser visível.

iii. Conjunto de ferramentas ou geoprocessos baseados, de um modo geral, nos mesmos pressupostos matemáticos, métricos (métrica euclidiana), geoestatísticos, lógicos, topológicos e gráficos (*layouts*, *labelling*, *simbology*) que permitem manipular a informação espacial da maneira mais adequada, tendo em vista a obtenção dos *outputs* finais.

Enfim, na extensa lista de *software* (Tabela IX) que combina a capacidade de manipulação de informação geográfica com os princípios da FSF ou OSI, apesar das muitas diferenças encontramos um número reduzido de *software* SIG que adoptam outras convenções, convergência e homogeneidade, elementos que fazem parte da identidade de um SIG e que agilizam a familiarização com a interface gráfica. O universo das TIG encontra-se actualmente em constante expansão, um fenómeno sem precedentes que deve ser visto pelo operador SIG, não com receio ou inércia, mas como uma oportunidade de enriquecer e alargar as suas capacidades para tratamento (interpretação e análise) da informação geográfica.

[64] O complexo problema do posicionamento é absolutamente relevante para quem trabalha com informação espacial, que, por esta razão, tem de ter uma geolocalização associada. Assim, qualquer *software* SIG representa informação espacial relativa a um referencial.

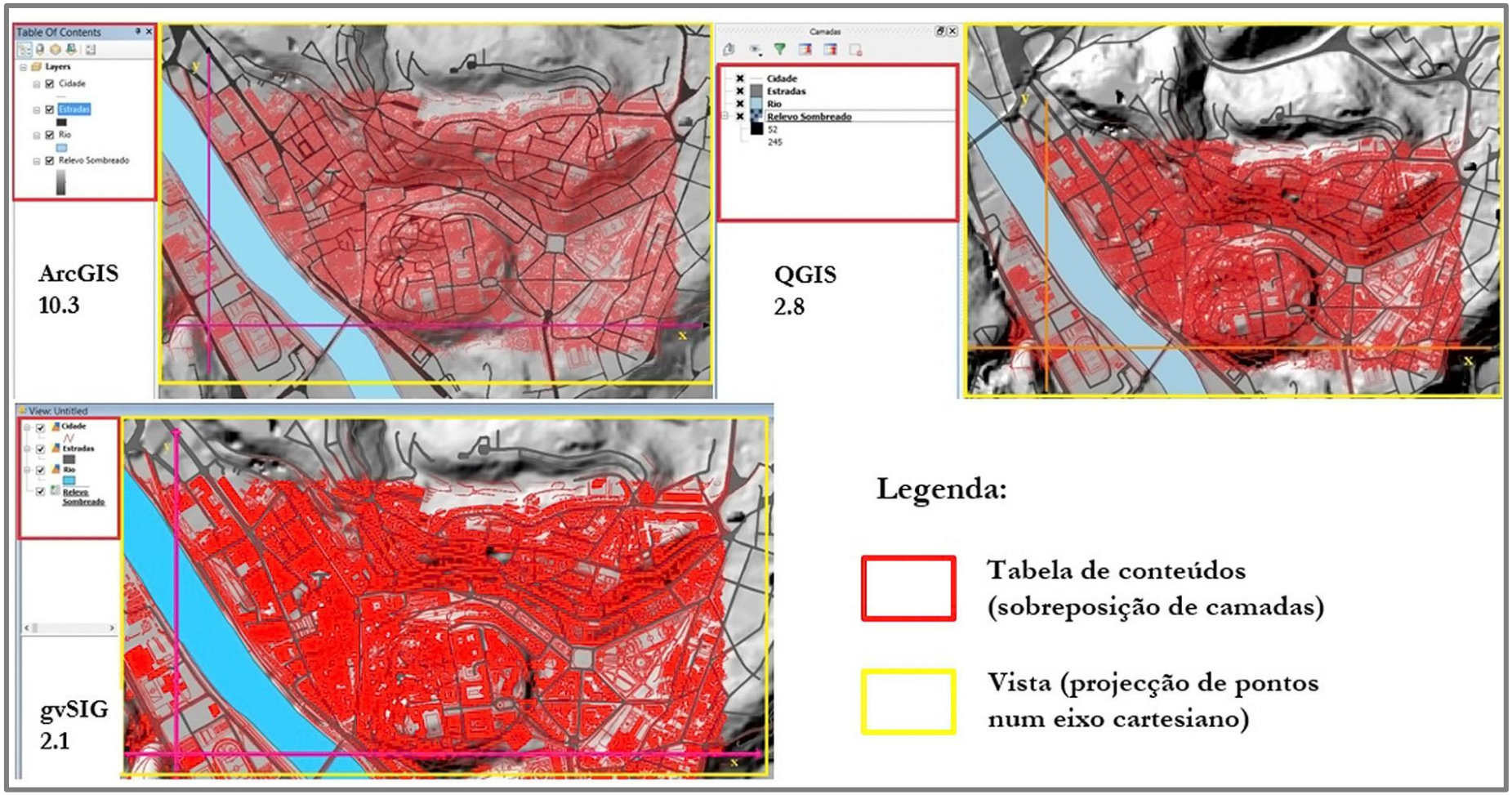


Figura 10 - As duas componentes essenciais da estrutura geral da interface de um *software* SIG: tabela de conteúdos para organização dos temas segundo a lógica da sobreposição de camadas; vista ou área de visualização, onde se visualiza a dimensão espacial dos temas em análise através da projecção dos seus pontos num eixo cartesiano referente ao sistema de referência/projecção implementado.

Hoje, são inúmeros os projectos que desenvolvem SL/CA para aplicação em áreas afins aos SIG, Detecção Remota, CAD e SGBD (Chen *et al.,* 2010; Cosme, 2012; Dobesova, 2013; Steiniger & Hunter, 2013)[65]. A Tabela IX (listagem completa no Anexo C) é um inventário que comprova a grande diversidade de *software* SL/CA vocacionado para operar nos campos de actuação das TIG.

Tabela IX - Inventário de SL/CA para aplicação na área das TIG e disponível para *download* – quadro completo no Anexo C

Nome	Versão actual	Endereço URL	Sistema Operativo	Linguagem programação
SIG *Desktop*				
GRASS GIS	7.0	http://grass.osgeo.org/	Mac OS, Windows, Linux	C, Python
gvSIG	2.2	http://www.gvsig.com:9090/pt	Mac OS, Windows, Linux	Java
ILWIS	3.8	http://52north.org/	Windows	?
OpenJUMP	1.8.0	http://openjump.org/	Mac OS, Windows, Linux	Java
QGIS	2.10.1	http://qgis.org/en/site/	Mac OS, Windows, Linux	C++
SAGA GIS	2.2.2	http://www.saga-gis.org/en/index.html	Windows, Linux	C++
Bibliotecas de algoritmos geoespaciais e *toolbox* incorporáveis em *software* SIG *desktop*				
GDAL/OGR	2.0.1	http://www.gdal.org/	Windows, Linux	C/C++
GEOS	3.5.0	https://trac.osgeo.org/geos/	MacOS, Windows, Linux	C++
Sextante	?	?	?	Java
Sistema de Gestão de Base de Dados				
PostgreSQL PostGIS	9.4.5 2.2.0	http://www.postgresql.org/	Mac OS, Windows, Linux	Java, C
Web SIG				

[65] Prova disso são as várias plataformas *online* que se apresentam como repositórios de informações sobre maior parte dos programas que aliam as capacidades de tratamento e manipulação geográfica aos fundamentos do SL/CA:

i. Freegis (http://freegis.org/, acedido em 16 Maio, 2015): este *website* organiza projectos relacionados com *software* e dados espaciais e suas documentações, de acordo com a sua aplicação (SIG *desktop*, WEBSIG, SGBD, SIG para telemóvel ou bibliotecas), linguagem de programação, licença, tipos de ferramentas, *standards* e estado de actividade. Segundo Chen *et al.* (2010), nele constam mais de 300 programas considerados como relevantes.
ii. SourceFourge (http://web.sourceforge.com/, acedido em 16 Maio, 2015): neste endereço constam todos os SL/CA, é uma plataforma que mantém e fornece os códigos para *download*.
iii. OSGeo (http://osgeo.org/, acedido em 16 Maio, 2015): o *website* da OSGeo foi criado com o intuito de prestar suporte e criar SL/CA de elevada qualidade, no domínio dos SIG.
iv. GISwiki (http://en.giswiki.next/wiki/Category:Software, acedido em 16 Maio, 2015): lista de software SIG, tanto proprietário como SL/CA.
v. A Wikipedia, no seu motor principal, disponibiliza também uma lista dos SIG (SL/CA) disponíveis (http://en.wikipedia.org/wiki/List_of_GIS_*software*, acedido em 16 Maio, 2015), bem como a comparação de alguns pacotes (http://en.wikipedia.org/wiki/Comparison_of_GIS_*software*, acedido em 16 Maio, 2015).

GeoServer	2.8.0	http://geoserver.org/	MacOS, Windows, Linux	Java
MapGuide	3.0	http://mapguide.osgeo.org/	Windows, Linux	PHP, Java
MapServer	7.0.0	http://www.mapserver.org/	MacOS, Windows, Linux	PHP, Python, Perl, C#, Java
OpenLayers	3.10.1	http://openlayers.org/	MacOS, Windows, Linux	Java
Detecção Remota e Fotogrametria				
Opticks	4.12	http://opticks.org/confluence/display/opticks/Welcome+To+Opticks	Windows, Linux	
Orfeo ToolBox Monteverdi2	5.0.0 0.8.1	https://www.orfeo-toolbox.org/	Windows, Linux	C++
OSSIM	1.8.20	http://trac.osgeo.org/ossim/	Mac OS, Windows, Linux	C++

Em síntese, não acompanhamos, em absoluto, a ideia de que a existência de diferentes graus de intuitividade no *software* SIG é infundada, porém, tendo em conta as premissas referidas neste texto que defendem um conceito de SIG independente do *software*, julgamos que tais argumentos não são suficientemente fortes ou incontornáveis ao ponto de inviabilizarem um processo de migração para SL/CA. A automatização de procedimentos (inércia) e a dependência relativamente ao SP têm de ser combatidas, não por causa do SP, mas porque um funcionário tem de ter uma atitude pró-activa e crítica no trabalho, procurando alternativas cada vez mais eficientes para os seus problemas, não apenas para benefício próprio, pessoal (enriquecimento científico e técnico) mas em prol da eficiência de recursos e da boa gestão da instituição que representa. Por outro lado, a questão essencial está em perceber qual é o *software* SIG que garante melhores resultados, dentro dos parâmetros previamente discutidos e estabelecidos pelo organismo para avaliar as opções de *software* disponíveis. Assim, apesar da grande diversidade de *software* expressa na Tabela IX[66], qualquer operador SIG de um determinado organismo tem a obrigação (caso haja um período de adaptação ou formação) de operar qualquer um destes *software*, desde que este lhe proporcione ferramentas e modos de manipulação de informação geográfica de qualidade.

[66] **Ver Anexo C.**

2. *Software* Livre e/ou de Código Aberto – uma visão enegrecida ou crónicas de uma narrativa imprecisa e distorcida?

> *"O grande problema que existe não tem a ver com aspectos legais ou políticos, mas antes com o nível de conhecimento e experiência de que as organizações públicas dispõem sobre as soluções informáticas existentes."*
>
> *In* OSEPA (2012) [67]

Apesar dos seus recentes e significativos desenvolvimentos, o SL/CA continua a ser fustigado pelo desconhecimento crónico que o classifica segundo um conjunto de características que não se coadunam com a realidade, mitos que revelam a falta de conhecimento que o público tem dele, facto do qual resulta a deturpação das suas reais vantagens e desvantagens.

A Figura 11 demonstra, de modo figurado, as dificuldades de comunicação entre os constituintes de um projecto de SL/CA e aqueles interessados em usufruir das suas potencialidades, evidenciando deficiências inerentes à transmissão do conhecimento relacionado com SL/CA. Problemas de comunicação que se se alastram à comunidade, em geral, e que generalizam, não só o desinteresse e a falta de conhecimento, mas também o medo, incerteza, inércia e dúvida (OSEPA, 2012b), algo que foi comprovado pelos dados recolhidos e pelos resultados obtidos nas respostas no inquérito que elaborámos e disponibilizámos para preenchimento público[68].

Com o intuito de contornar a falta de conhecimento sobre SL/CA[69] e estimular a sensibilização para o seu uso, tomámos por boa, a opção de problematizar e discutir alguns dos mitos mais frequentes, descortinando a sua veracidade ou inconsistência com a realidade, no sentido de

[67] No original, "*The main problem that exists is not the legal and political framework, but the level of knowledge and experience of the public organizations about the available software solutions that exist.*" *In* OSEPA (2012) – "OSEPA CP3 expert survey conclusions". 58 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

[68] Na discussão dos mitos, vantagens, inconvenientes e oportunidades foram usados os resultados do inquérito difundido no âmbito da realização deste trabalho de dissertação. Assim, a argumentação empregue neste tópico resulta também de experiências reais e vividas sobre o SL/CA na APP.

[69] Segundo a OSEPA (2012c), a ignorância e a desinformação são um dos factores que condicionam fortemente a adopção de SL/CA.

promover as principais vantagens e clarificar quanto às desvantagens associadas à utilização de SL/CA.

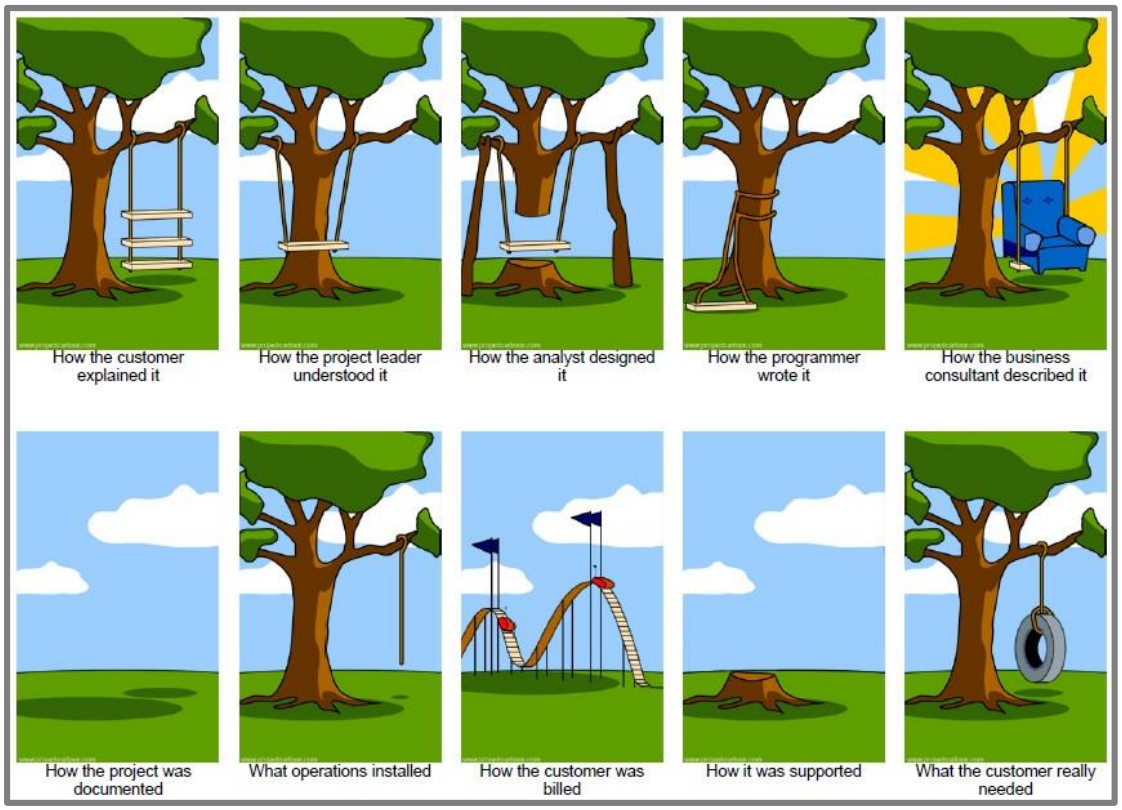


Figura 11 - Uma linguagem metafórica para retratar algumas das incompreensões e falhas de comunicação associadas a um projecto relacionado com SL/CA. Fonte: Wise (2012), p. 67.

2.1 Aparente gratuitidade do *Software* Livre e de Código Aberto

Gay (2002) refere-se à posição de Stallman que defende que o *software* deve ser livre, o que significa que não é, necessariamente, gratuito. Portanto, SL não está necessariamente isento de custos, como defende também Souza (2006), sendo, aliás, curioso constatar que é o próprio criador da FSF que encoraja as pessoas a cobrarem o que possam ou o que achem justo pela distribuição de SL – a GPL não permite a venda explícita de *software*, mas o direito de vender cópias é parte da sua definição (https://www.gnu.org/licenses/gpl-faq.pt-br.html, acedido em 9 Abril, 2015; Steiniger & Bocher, 2009) [70]. Não obstante, a maioria dos projectos, disponibiliza o seu *software* sem exigir o

[70] É dada a qualquer um a permissão de vender cópias, mas não é possível exigir-se, a qualquer um que receba uma cópia, o pagamento de uma taxa ou o acto de notificar o desenvolvedor. Por outras palavras, se um desenvolvedor A, vender uma cópia a um utilizador B, este último tem permissão para redistribuir esse programa como bem entender (desde que cumpra os requisitos da licença), sem que qualquer outro utilizador, que receba o *software* do utilizador B, tenha de pagar algo ou notificar o desenvolvedor A. Inclusive, Stallman refere que ele próprio vendeu as primeiras

pagamento de qualquer valor, convidando os utilizadores a fazerem uma doação facultativa ao projecto.

Neste sentido, Glynn *et al.* (2005) realçam a atractividade do baixo custo do SL/CA; DiBona (2006) refere que em muitos casos "…*free things are still cheaper than expensive things*" (op. cit. p. 36); um dos entrevistados de Marsan & Paré (2013) salienta que "…*the cost of OSS is a bargin in comparison*[71]*"* (op. cit. p. 735). Em consequência destas reflexões propostas pelos autores citados, parece legítimo concluir que soluções em SL/CA revelam elevada competitividade em termos de custo quando confrontados com soluções proprietárias.

Ainda assim, a gratuitidade é uma ilusão; como referimos antes, não existem soluções sem custos. Em alguns casos, estes podem mesmo exceder os custos da aquisição/manutenção de licenças de SP (Marsan & Paré, 2013).

A literatura da especialidade e as boas práticas tornadas públicas revelam que, para o SL/CA ser viável, tem de haver fluxos monetários originados, não pela venda do *software*, mas sim pela contratualização de serviços de formação, suporte e outros (OSEPA, 2012b) – custo económico da solução. Por exemplo, Wasserman & Capra, (2007) *apud* Hauge (2010) relatam que 50% do código fornecido à comunidade é pago aos desenvolvedores; por aqui se percebe a inevitabilidade da necessidade de verba para suportar custos logísticos e trabalho, e para gerar mais-valias no desenvolvimento das soluções de SL/CA.

Para além disto, a contabilização rigorosa dos custos, não pode deixar de tomar em linha de conta outros custos para além da mera contabilização dos gastos com a aquisição deste tipo de encargo; existem outras dimensões que, a médio/longo prazo, se traduzem em custos financeiros avultados, fruto da quebra de eficiência (porque não se avaliou correctamente a solução SL/CA implementada). Isto pode ocorrer por diversas ordens de razões, entre as quais se destaca a insatisfação do pessoal ao serviço (por inércia, automatização de procedimentos já adquiridos e dependência relativamente ao SP, desconforto com a nova solução que obriga a criar nova empatia com o utilizador, desempenho lento do SL/CA perante determinadas tarefas, etc.) com a solução adoptada (factores organizacionais), do apoio da comunidade (factores externos), ou da necessidade de produzir novas ferramentas (factores técnicos).

O planeamento do processo de migração é fundamental se se quiserem minimizar efectivamente as despesas. Esta tarefa envolve também um grande esforço (custo), o de aprender a entender as

cópias do sistema operativo GNU, referindo que distribuir SL é uma forma de angariar fundos para o seu contínuo desenvolvimento (Gay, 2002).

[71] O entrevistado refere-se à comparação entre o SL/CA e o SP.

novas componentes, conhecimento esse que permitirá escolher componentes de qualidade que respeitem as especificações do trabalho da organização (Chen *et al.*, 2008 *apud* Hauge, 2010).

Enfim, não havendo soluções de custo zero, torna-se imperativo proceder a um exercício onde a ponderação individual e contextual (caso a caso) do TCO e ROI quer para as soluções SL/CA quer para as de SP, passa a desempenhar papel nuclear.

2.2. A Catedral que se ergue no centro do Bazar

Anteriormente, fez-se referência à transversalidade entre SP e SL/CA (DiBona, 2006). Essa alusão não foi despropositada. A transferência dos conhecimentos sem perdas faz-se em liberdade, mas normalmente mediante pagamento (isto porque, grande parte das vezes, quem tem interesse num determinado recurso informático, não tem conhecimentos na área o que o torna dependente de outrem). Isto fez com que o SL/CA evoluísse de um modo que o tornou um modelo de negócio que visa a obtenção de lucro (Mahony & Naughton, 2004; Fitzgerald, 2006). São vários os autores que focam aspectos relacionados com esta génese comercial e economicista, aspectos como:

i. Os projectos de SL/CA, sobretudo os mais activos, recebem, por vezes, ajuda de companhias/empresas que tentam tirar partido das suas vantagens, convertendo-as em benefícios para a organização (Bonaccorsi *et al.,* 2007 *apud* Hauge *et al.*, 2010).

ii. São várias as companhias que patrocinam estrategicamente indivíduos que influenciam as comunidades, em consonância com os seus objectivos (Dahlander & Wallin, 2006 *apud* Hauge *et al.,* 2010; Dahlander & Magnusson, 2008 *apud* Hauge *et al.*, 2010).

iii. Por vezes, como referem Dalle & Rousseau (citados por Hauge *et al.*, 2010), assiste-se à transferência de produtos de código aberto desenvolvidos em meios académicos para produtos comerciais derivados (que beneficiam da flexibilidade do SCA referida na secção anterior).

iv. Não é incomum encontrar companhias com estratégias híbridas – lembre-se o termo *dual licensing* (West, 2003 *apud* Hauge *et al.*, 2010).

Tudo isto para dizer que o SL/CA não é somente benfeitoria altruísta que tem em vista a partilha do conhecimento, e que se insere, portanto, nas regras do mercado capitalista. Ainda assim, para quem instala SL/CA, a independência relativamente aos interesses de qualquer empresa é

assegurada desde que exista o conhecimento necessário para adaptar/actualizar o *software* de acordo com as suas necessidades, caso isto não seja possível, não há fuga a encargos ou condicionalismos[72].

2.3. Disponibilização do código-fonte - um trunfo para o desenvolvimento técnico e económico?

Na sequência do exposto, o SL/CA recorre à disponibilização do código – raiz do programa como arma fundamental, cativando potenciais clientes/utilizadores com um modelo de desenvolvimento flexível, criativo e democrático. Esta forma de proceder desperta também o interesse das companhias que tentam pensar em maneiras de a rentabilizar.

A transparência concede a oportunidade de estudar o programa, identificando e corrigindo mais facilmente quaisquer erros (Krogh & Spaeth, 2007). No entanto, apesar disto poder indiciar uma elevada qualidade pela detecção e correcção rápidas dos *bugs*, é difícil afirmar que existe uma relação causal entre o *software* ser de código aberto e a sua eficácia, eficiência, qualidade e valor (Fuggetta, 2003).

O livre acesso ao código-fonte é, de facto, uma das bandeiras estruturantes do SL/CA. A capacidade de modificar e adaptar o programa é uma vantagem, considerada até como uma mais-valia quando analisada em termos de retorno do investimento. Porém, o sucesso dos projectos está dependente de mais variáveis, para além do facto do código ser legível, de tal modo que as relações causa-efeito esboçadas pela Figura 12 não são passíveis de validação com um carácter verdadeiramente científico.

Objectivamente, na avaliação dos parâmetros envolvidos na implementação e sucesso do processo de migração para SL/CA defende-se a inclusão de um amplo espectro de factores e de preocupações que suplantam a questão da abertura do código (Glynn *et al.*, 2005; Qu *et al.*, 2011); Li *et al.*, 2012; Bouras *et al.*, 2013; Marsan & Paré, 2013; Sarrab & Rehman, 2014).

Em segundo lugar, para a maior parte dos utilizadores finais (sem conhecimentos em programação), tal aspecto é absolutamente irrelevante.

[72] Por exemplo, a Empresa A, está de algum modo ligada ao *Software* X (que é de código aberto); sabe-se que quem usa o *Software* X necessita de algumas actualizações em determinadas ferramentas; logo, a dita empresa não tem interesse em promover junto da comunidade o desenvolvimento das ferramentas que os utilizadores necessitam; provavelmente, preferem, eles próprios, desenvolvê-las, cobrando um preço justo por esse serviço. Este é um exemplo hipotético de como os utilizadores podem continuar a depender dos intentos de qualquer companhia.

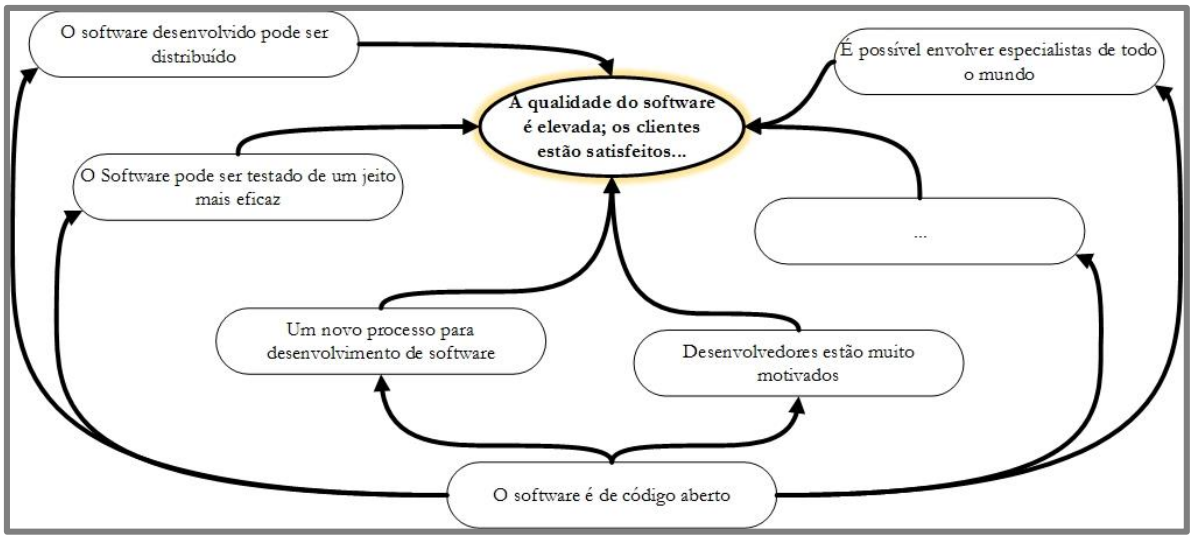


Figura 12 - A importância do código aberto – diagrama de causa efeito. Fonte: Fuggetta (2003), p. 84.

Em terceiro lugar, a possibilidade de estudar e perceber o modo de funcionamento do programa é o que mais cativa os investigadores. Ainda que esta possibilidade seja uma vantagem aliciante, facilitadora da aprendizagem daqueles que dão os primeiros passos na informática e programação, este não deixa de ser um argumento propagandista que nos induz ao facilitismo.

Não deixa de ser curioso constar que, como nos diz Seltzer (2006), a lei dos direitos de autor não protege funções, métodos ou procedimentos que a expressão intelectual protegida implementa, logo, mesmo com o código fechado, um programador é livre de estudar interfaces e funcionalidades, livre de operar e substituir esse código com o da sua autoria.

Na realidade, recorrendo a um exemplo no âmbito dos SIG, sem se ter acesso ao código-fonte, com alguns conhecimentos estruturais sobre o modelo de dados vectorial (pontos com coordenadas rectangulares ou geográficas armazenados em vectores), geodesia e georreferenciação (transformação de coordenadas) e de programação, a construção de um algoritmo de transformação do sistema de referência espacial de uma *shapefile,* é uma tarefa trivial. O entendimento do modo como a ferramenta funciona é, antes de mais, lógico e matemático, só depois vem a componente de análise ou incrementação de um bloco de código. Ou seja, com o código fechado, somos obrigados a pensar, ao invés, com o código aberto, podemos cair na tentação de o executar sem pensar na sua estrutura, isto é, executamo-lo sem saber a lógica processual que se encontra por trás da gramática do próprio código.

Neste sentido, há quem afirme que o SP, nomeadamente SIG, não é assim tão secreto e impenetrável, pois, de facto, grande parte dos algoritmos incluídos nos SP mais populares foram

desenvolvidos nos anos 70, e estão publicados, tal como nos mostram Burrough & McDonnell (2000), Matos (2008) e as várias referências bibliográficas que constam do *help* destes *software*.

Assim, entendemos que a grande vantagem da abertura do código se pode explicar segundo a seguinte ideia: na implementação de um modelo de adopção de SL/CA, é legítimo que os administradores se questionem se o SL/CA que equacionam adoptar é eficaz numa perspectiva matemática. Existe a hipótese de se encontrarem algoritmos similares/análogos que, ao contrário do expectável, apresentam resultados diferentes. A avaliação rigorosa de possíveis inconformidades matemáticas (qualidade das ferramentas) torna-se possível com a análise do código-fonte (OSEPA, 2012b).

Visto que a APP tem de privilegiar a transparência das suas decisões em matéria de ordenamento do território, o SL/CA deve ser a primeira alternativa a ser considerada.

A maleabilidade do programa seria também uma mais-valia para estes órgãos (deixa de estar dependente do fornecedor), contudo, em Portugal, em determinadas instituições como as câmaras municipais, seria incomportável contratar pessoal qualificado apenas para a adaptação do SL/CA, o que diminui a importância desta vantagem. Ou seja, cada órgão deve ponderar este factor segundo um peso que se coadune com os benefícios que poderão vir a tirar com o código aberto.

2.4. Comunidade! Comunidade! Comunidade!

Qu *et al.* (2011) salientam a importância que as redes têm no processo de adopção de SL/CA, ou seja, os contactos com a comunidade de utilizadores de SL/CA e a rede de outras organizações que usam este tipo de *software*, é um factor potenciador da migração. O acesso a uma comunidade de utilizadores, especialistas e desenvolvedores e a possibilidade de receber suporte dessa mesma comunidade são apontados como uma das grandes vantagens do SL/CA (Krogh & Spaeth, 2007; OSEPA, 2012b).

Embora a abnegação (que implica partilhar algo de valor para outros, desenvolver esforços para um bem maior, ajudar o outro, crença pessoal no SL/CA e melhoria da qualidade) seja uma característica base destes movimentos, que faz com que os indivíduos participem na comunidade (Baytiyeh & Pfaffman, 2010), esta é, antes de mais, um potencial a que os desenvolvedores e utilizadores recorrem para fazer avançar novas versões num curto intervalo de tempo[73]. Como é

[73] O Ubuntu é um dos sistemas operativos Linux mais populares - http://www.ubuntu.com/- , a cada 6 meses é disponibilizada uma nova versão, enquanto que o sistema operativo proprietário de referência no mercado é lançado

expectável, as companhias promovem formas de usufruir destas redes, colaborando tendo em vista o fornecimento de um *whole product* (Feller *et al.*, 2008 *apud* Hauge *et al.*, 2010). Porém, este possante processo de melhoria continuada não tem necessariamente apenas aspectos positivos, uma vez que a actualização para uma nova versão exige sempre uma adaptação às mudanças na interface (no caso do sistema operativo, a exigência de criação de cópias de segurança dos ficheiros é um processo que consome bastante tempo), bem como (em algumas situações) não está garantida a interoperabilidade entre os formatos anteriores e os implementados pelas novas versões (Kovács *et al.*, 2004). Este é outro dos factores que deve merecer destaque quando se comparam soluções, na medida em que, sendo impossível actualizar constantemente as versões do *software*, temos de estar certos que o projecto disponibilizará suporte e actualizações para a versão que usamos actualmente durante um período aceitável.

Independentemente disto, a comunidade garante suporte aos utilizadores, rastreia e corrige erros, avalia a segurança do *software*. Ela é, sem margem para dúvidas, uma mais-valia: por um lado, enquanto no SP, havendo um *bug*, temos de ficar à espera de uma nova versão ou de um *service pack* de actualização (esperando que não haja custos), com o SL/CA, o problema poderá ter solução em poucas horas/dias[74]; por outro lado, a inteligência colectiva, à qual se vincula a lógica dos muitos olhos vêem mais, melhor e mais depressa, impede que qualquer 'terrorista informático' modifique o *software* com propósitos menos nobres ou por incapacidade técnica (Raymond, 2001).

Mas as comunidades não são entidades uniformes nem homogéneas, algumas são mais exigentes, pelo que pode ser desafiante para os desenvolvedores ver aceites os seus contributos, outras são menos exigentes (Hauge,*et al.*, 2010). A qualidade do projecto depende necessariamente do número (e qualidade) de desenvolvedores que contribuem realmente para a comunidade.

O seguinte exemplo é um caso pontual de ineficácia da comunidade, neste caso específico, associada ao projecto QGIS (http://qgis.org/). Patriarca *et al.* (2014a) desenvolveram um exercício que consistia no corte topológico da Carta de Risco de Incêndio Florestal (CRIF) de 2011 (para o território continental português), em formato *shapefile* (documento com 1,33 GB e, aproximadamente 3 500 000 *features*) segundo os limites administrativos do distrito de Coimbra (depois de efectuado um processo de agregação dos limites das freguesias considerados na Carta Administrativa Oficial de Portugal – CAOP de 2014), com o propósito de comparar o desempenho

entre espaços temporais mais alargados. Mas, as versões anteriores não são esquecidas, sendo garantido para cada versão um suporte de actualizações de 5 anos. O QGIS (http://qgis.org/en/site/) é um outro projecto que disponibiliza novas versões a cada 3 meses.

[74] O período de resolução do problema está dependente, como é óbvio, da natureza do problema.

dos algoritmos análogos disponibilizados por três *software* de referência, na execução desta tarefa (SP ArcGIS 10.2, gvSIG 2.0 e QGIS 2.2).

Os resultados obtidos revelaram que o QGIS, em todos os testes realizados, levou mais do que 30 minutos a devolver o resultado pretendido. Considerando estes tempos pouco aceitáveis, porque evidenciam uma fragilidade de eficiência relativamente ao SP também sob escrutínio, comunicou-se esta limitação aos desenvolvedores do projecto. Depois de 5 actualizações, repetiu-se exactamente o mesmo procedimento com a versão 2.8 (Figura 13) [75].

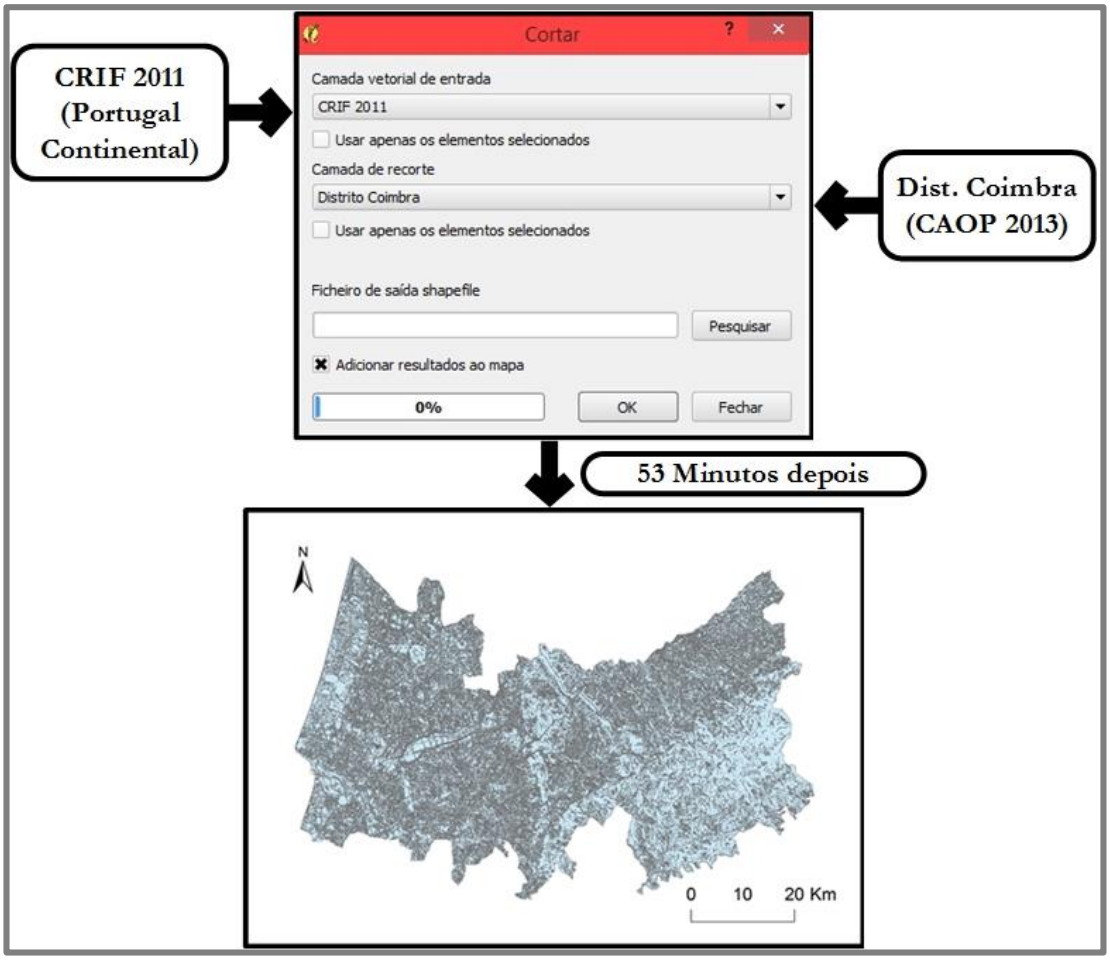


Figura 13 - Procedimento e resultado do corte topológico da CRIF (2011) – vectorial – no QGIS (Versão 2.8.1), com base no limite administrativo do distrito de Coimbra (depois de concretizado um processo de agregação das freguesias – fonte CAOP, 2014).

[75] Algoritmos/comandos utilizados: Menu Vector > Ferramentas de geoprocessamento > Cortar > Configurar e executar.

O facto de o algoritmo de corte ter demorado 53 minutos a executar a tarefa demonstra que o problema não foi solucionado, o que é uma evidência de que, mesmo que pontualmente e em raras situações, a comunidade pode ser pouco eficaz na resolução de determinados erros.

Para concluir, não podemos deixar de realçar que a existência de uma comunidade que presta suporte não é exclusiva do SL/CA. O SP também possui uma comunidade de utilizadores bastante activa, sendo bastante simples encontrarem-se soluções, isto é, embora a comunidade dos projectos de código aberto seja muito mais organizada, não se pode dizer que esta seja uma vantagem exclusiva do SL/CA.

2.5. Certificação e creditação de *Software* Livre e de Código Aberto

Existem órgãos que certificam e garantem a qualidade do *software*. No que diz respeito aos SIG, destacam-se o OGC (http://www.opengeospatial.org/, acedido em 18 Agosto, 2015), cuja finalidade é a definição de padrões abertos (*open standards*) e interoperáveis dentro dos SIG, e o OSGeo (http://www.osgeo.org/, acedido em 18 Agosto, 2015), criado para fornecer suporte e desenvolver sistemas de alta qualidade, em código aberto, para aplicações geoespaciais.

Para além da FSF e OSI (referências do sector que têm como preocupação genérica o licenciamento e salvaguarda do SL/CA), Kemp (2009) refere outros organismos que têm como meta fazer doutrina o uso de SL/CA e fortalecer a argumentação em seu torno, como o *Software Freedom Law Center* (SFLC) - http://www.*software*freedom.org/ (acedido em 11 Agosto, 2015) – que providencia representação legal e outros serviços relacionados com a lei para proteger e desenvolver o SL/CA.

Em Portugal, merecem referência organismos como a ESOP (http://www.esop.pt/Default/pt/Homepage, acedido em 11 Agosto, 2015), uma associação empresarial que representa as empresas portuguesas que se dedicam ao desenvolvimento de *software* e à prestação de serviços baseados em tecnologias de código aberto, privilegiando a qualidade, a inovação, a interoperabilidade, as normas abertas e a independência de plataformas tecnológicas. Também a ANSOL (Associação Nacional para o *Software* Livre) merece nota de destaque, por se tratar de uma associação sem fins lucrativos que tem como fim a divulgação, promoção, desenvolvimento, investigação e estudo da informática livre e das suas repercussões sociais, políticas, filosóficas, culturais, técnicas e científicas (https://ansol.org/, acedido em 19 Novembro, 2015), assim como a AMA (Agência para a Modernização Administrativa) a quem foram atribuídas

competências ao nível da regulamentação relacionada com a interoperabilidade e com a gestão de processos de aquisição de *software*.

2.6. Suporte, manutenção e garantias

De acordo com Glynn *et al.* (2005), a ausência de um vendedor tradicional (que garanta assistência) é um inibidor da migração, porém, embora possa ser um argumento justificável em alguns casos, na maior parte das vezes dizer-se que se verifica a ausência de assistência técnica adequada e profissional é incorrecto, ainda que, em alguns ramos de actividade (como pode ser o caso de alguns órgãos da administração pública), segundo os estudos conduzidos no âmbito do projecto OSEPA, as companhias que fornecem suporte sejam pequenas ou médias empresas sem capacidade para satisfazer a procura, recordando o velho "dilema do ovo e da galinha" – a APP não pode adoptar SL/CA sem suporte, as companhias que prestam este serviço não podem emergir porque o SL/CA não é usado (OSEPA, 2012b).

No entanto, para além do *backup* da comunidade, existem várias empresas (no mundo e em Portugal) que oferecem este tipo de serviços. As empresas que constituem a ESOP (Figura 14) são exemplo disso mesmo.

Figura 14 - Empresas que constituem a ESOP. Fonte: http://www.esop.pt/Default/pt/Associados (acedido em 9 Maio, 2015).

Este suporte, por ser profissional, tem de ser contratualizado, o que implica encargos e pode anular os benefícios que o baixo custo do *software* traria à partida. Dependendo do tipo de pessoal e do serviço em que se utiliza, é provável que se tenha de estabelecer contratos com empresas, capacitadas para dar formação ao órgão que inicia o processo de implementação de SL/CA. Em alguns casos, a probabilidade de não se registarem poupanças reais não deve ser ignorada.

Admite-se que o processo de migração não seja muito bem-sucedido sem este tipo de contratação. Para além disso, tendo em conta o relatório da OSEPA (2012b), para um determinado ramo específico, deve-se verificar cuidadosamente se existem empresas que prestem assistência em determinada especialidade.

Relativamente à documentação disponibilizada, ela é geralmente de boa qualidade, muito embora a incompletude dos documentos seja recorrente em alguns projectos SIG (Steiniger & Bocher, 2009). Por exemplo, são várias as obras publicadas que constituem manuais de apoio à utilização de SL/CA (Sherman, 2008) não apenas em SIG como também em áreas afins como a Estatística Espacial e as Bases de Dados Espaciais – destacam-se programas como o R Statistics (Quick, 2010; Dorman, 2014; Gerrard & Johnson, 2015), o PostgreSQL/PostGIS (Momjian, 2001; Matthew & Stones, 2006; Riggs & Krosing, 2010; Smith, 2010; Obe & Hsu, 2011 e 2012; Chauhan, 2015), o MySQL (Nixon, 2009; Yank, 2009; Davies, 2010; Lukaszewski, 2010), o GRASS (Neteler & Mitasova, 2008), o QGIS (Westra, 2014; Tom, 2015) ou o Open Layers (Gratier *et al.*, 2015), etc., aos quais se juntam os manuais de utilizador disponíveis nas páginas do projecto.

Finalmente, sobre as garantias, resta esclarecer que nenhum fornecedor de *software* pode ser responsabilizado legalmente por qualquer falha no seu programa, uma vez que, ao contrário de um electrodoméstico, quando se compra *software*, aplica-se o direito de autor e não o direito comercial, razão pela qual os produtores se desresponsabilizam de qualquer mau funcionamento (APDSI[76], 2004). Por outras palavras, a licença do *software* é normalmente incorporada por aquilo que Laurent (2004) nomeia como *Warranty disclaimers*, em vez de uma *express warranty*. Esta, está vinculada a qualquer item particular que possa ser comprado, se esse objecto não tiver as funcionalidades que o fornecedor alega que ele tem, o cliente pode exigir a troca do produto ou a devolução do seu valor. Ao invés, a primeira é uma declaração que define que o fornecedor de *software* não está vinculado a quaisquer direitos de garantia do cliente.

Assim, como explica Laurent (2004), as *warranty disclaimers* podem gerar novas oportunidades de negócio por permitir o estabelecimento de contratos, em que se define como é que os clientes

[76] Associação para a Promoção e Desenvolvimento da Sociedade da Informação.

obtêm suporte para um produto que não tem garantia, serviço que será alvo de remuneração em benefício dos desenvolvedores ou de outros delegados dessa tarefa.

2.7. Robustez técnica do *Software* Livre e de Código Aberto

O SL/CA não é mágico, não trabalha à velocidade da luz, muitas vezes as equipas são pequenas, tornando a competição difícil (DiBona, 2006). Os dados obtidos ao longo desta investigação registam exemplos de utilizadores que garantem ter tido más experiências com ferramentas de código aberto. Inclusive, há quem considere que o SL/CA é óptimo numa perspectiva de investigação individual, porém inadequado e complicado para um serviço com clientes. Logo, há relutância em utilizar novas ferramentas, sobretudo quando as que existem (proprietárias) resolvem os problemas com facilidade.

Chen *et al.* (2008), *apud* Hauge *et al.*, (2010) e Ven & Mannaert (2008) *apud* Hauge *et al.*, (2010) argumentam que a disponibilidade de conhecimento é um factor decisivo no processo de migração, por um lado auxilia na escolha da tecnologia e, por outro, é necessário para proceder às modificações e correcções que recorrentemente precisam ser efectuadas. Estes autores identificam desafios e alertam para a obrigatoriedade de promoção de estratégias para lidar com estas modificações, de maneira a minimizar os custos.

A falta de qualidade/robustez técnica não é de todo um mito, é uma realidade (Stamelos *et al.*, 2002 *apud* Hauge *et al.*, 2010), tanto que em determinadas ocasiões se torna difícil encontrar um 'SL/CA campeão', factor inibidor da tomada de decisão no sentido da migração (Glynn *et al.*, 2005). Ainda assim, devem-se evitar generalizações. A Figura 13 demonstra o mau funcionamento de uma ferramenta de geoprocessamento que, por um lado, não inviabiliza o trabalho com o programa (todos têm os seus erros e as suas insuficiências, mesmo o SP), por outro, não quer dizer que não existam outros SL/CA capazes de resolver, de modo eficiente, o pretendido. Aliás, os investigadores da OSEPA consideram que o SL/CA detém elevada qualidade e o seu código-fonte pode ser enriquecido, o que se traduz por melhorias na própria aplicação (OSEPA, 2012b). Há alguns indicadores que apontam nesse sentido, nomeadamente: comunidade de suporte, na qual participam programadores que melhoram o programa gradualmente; obras (como aquelas referidas em secção precedente deste trabalho) e documentação que revelam as potencialidades do *software*; casos de sucesso.

No que concerne aos SIG, o grau de complexidade com que um SL consegue lidar depende do nível de maturidade do projecto. O mais recorrente é encontrar-se um programa adequado para um problema específico, mas que evidencia limitações quanto a outro tipo de questões.

Por isso, aquando da tomada de decisão sobre a solução a manter/implementar, os decisores devem aferir a qualidade técnica das soluções, avaliação suportada pela apresentação de testes comparativos (ponderação da eficácia, fiabilidade, performance e grau de desempenho dos programas, bem como a produtividade dos trabalhadores).

2.8. O processo de migração

A migração é, sem dúvida, um processo difícil e custoso (a diversos níveis), sobretudo se for realizado de forma isolada; a lógica colaborativa deve ser valorizada e posta em primeiro plano.

A sua preparação exige, desde logo, uma profunda reflexão sobre um largo espectro de factores e parâmetros, uma vez que se deve garantir que não haverá problemas de desempenho, dificuldades em migrar a informação das BD pré-existentes (assegurar interoperabilidade entre *software* e formatos), e que o desenvolvimento da solução terá continuidade no tempo (garante de suporte e assistência – período de vida). Só assim, com conhecimento, se abandonam medos e crenças não fundamentadas quanto ao desempenho e vulnerabilidade (segurança informática) das soluções de SL/CA.

Obviamente que, como referem Glynn *et al.* (2005), o nível de formação, o grau de empatia com a filosofia do SL/CA, o grau de cristalização, inércia e comodismo são factores determinantes, tornando o processo um problema que depende sobremaneira de factores de natureza individual, de competências mas também de personalidade.

Finalmente, conclui-se que a misticidade e as vantagens/desvantagens que conotam o SL/CA estão claramente condicionadas pelo tipo de organismo que o pretende adoptar e pela área técnica em que está envolvido (pois, aqui regista-se uma variação na maturidade dos diferentes programas e na disponibilidade de suporte e assistência profissional à solução livre). Neste contexto, é aconselhável, para cada caso de aplicação, a elaboração de uma análise que englobe os parâmetros que justificam a implementação de SL/CA ou a manutenção de soluções proprietárias – concepção de um modelo de adopção que conceba estratégias de valoração das alternativas em quatro dimensões fundamentais: financeira, técnica, organizacional e suporte externo. Essa avaliação

poderá, até, ter como ponto de partida, uma análise SWOT[77] geral como a apresentada na Tabela X.

[77] Do inglês *strenghts* (forças), *weaknesses* (fraquezas), *opportunities* (oportunidades) e *threats* (ameaças).

Tabela X - Sistematização das principais vantagens e desvantagens do SL/CA através de uma análise SWOT

Forças	→ Normalmente, não acarreta custos de aquisição ou licenciamento; → Capacidade de modificar e adaptar o *software* às necessidades pelo acesso ao código-fonte; → Suporte da comunidade (inteligência colectiva); → Órgãos credíveis que regulam e certificam; → Elevado índice de interoperabilidade (implementação de normas abertas); → Legislação que incentiva/obriga a adopção de SL/CA – apoio governamental e de instituições internacionais (UE); → Forte suporte por parte de companhias multinacionais; → Lançamento frequente de novas versões - melhorias; → Exige menos do hardware que as soluções fechadas; → Baseado em *standards* abertos; → Promove a cultura da legalidade; → Implementações com sucesso (Brasil, Espanha, Áustria, Alemanha, Portugal, Itália, etc.);	**Fraquezas**	→ Documentação incompleta ou pouco pormenorizada em alguns casos; → Probabilidade de haverem custos avultados com suporte, manutenção e suporte; → Falta de partilha de experiência e de dados empíricos; → Possíveis incompatibilidades com o hardware; → Em alguns casos, tendo em conta a maturidade do *software* e disponibilidade de documentação, só com a contratualização serviço de suporte e formação é que se pode tirar o máximo de partido da solução livre/código aberto; → Forte dependência relativamente ao nível de maturidade do *software* – um baixo nível traduz-se em fraca robustez e em dificuldades de utilização e aplicação eficiente; → Por vezes, exigem conhecimentos avançados de informática e programação – condicionado pelo nível de amigabilidade e capacitação da interface; → Casos de *software* incapazes (insuficientes; ineficientes) ou de baixa qualidade;
Oportunidades	→ Liberdade de pensar e de criar – ciência deve ser baseada no livre intercâmbio; → Directivas e projectos internacionais que fomentam a sua utilização através da incrementação de normas abertas, normalmente contempladas no desenvolvimento de SL/CA; → Novas oportunidades de negócio através da produção de *software* derivado (facilitado pelo acesso ao código-fonte) ou serviços – estímulos ao sector e à economia (competitividade melhora os produtos); → Abre portas de negócio para empresas capazes de prestar suporte a este tipo de *software*; → Elevado índice de interoperabilidade; → Reduz a dependência relativamente a fornecedores de *software*; → Possibilidade de adaptar/modificar o *software*; → Perfeitamente enquadrável no sistema de ensino – livre distribuição que permite aos alunos usarem em casa sem limitações; correcção de erros ou adaptação do programa como forma de aprendizagem; → Lançamento rápido de actualizações/novas versões – melhoria constante;	**Ameaças**	→ Desinteresse/inércia dos programadores em corrigir/prestar suporte ao SL/CA; → Tentativa de correcção de problemas por programadores de baixo nível; → Concorrência com *software* de muito boa qualidade; → Incompatibilidades entre formatos – com os proprietários ou com os das novas versões que vão sendo lançadas; → Lançamento rápido de actualizações/novas versões – mudanças constantes;

CAPÍTULO II – *SOFTWARE* LIVRE E DE CÓDIGO ABERTO, UMA SOLUÇÃO DE CONFIANÇA E FIABILIDADE

Uma das adversidades que se colocam à implementação de SL/CA em qualquer organização, pública ou privada, tem a ver com o reduzido nível de confiança que muitos administradores revelam ainda em relação ao SL/CA. Esta constatação, que se suporta numa fundamentação de base empírica, de conhecimento de terreno, leva-nos a admitir que, antes de se equacionar uma eventual migração, *a montante*, deve ser implementado um exercício de base analítica que permita elencar e anunciar as principais razões em relação às quais os gestores públicos devem estar sensibilizados no sentido de se proceder a uma adequada ponderação como suporte à tomada de decisão. Assim, o capítulo que agora se inicia debruça-se sobre os seguintes temas:

1. **Organismos, normas, formatos e protocolos, especificações e *standards***
    i. Demonstrar a fiabilidade do SL/CA, nomeadamente SIG, através da apresentação e caracterização de organismos, iniciativas, projectos e *standards* que, de um modo ou de outro, atestam a sua credibilidade e fomentam a sua qualidade, segurança e interoperabilidade.
2. **O panorama legislativo português e a obrigatoriedade de implementação de *software* de baixo custo**
    i. Revisitar os diplomas legais que obrigam os órgãos que compõem a APP a justificar todas as suas opções em matéria de aquisição de *software*.
    ii. Discutir as implicações dos parâmetros presentes nestes decretos que possam, de algum modo, permitir, às instituições públicas, contornar as obrigatoriedades estabelecidas pelas prerrogativas de base jurídica.
3. **O *statu quo* da adopção de *Software* Livre e/ou de Código Aberto SIG, no mundo e em Portugal**
    i. Demonstrar que o SL/CA é uma solução que tem vindo a crescer rapidamente em número de adeptos, o que, de algum modo, atesta a qualidade e a forte capacidade de

desenvolvimento endógeno das suas limitações ou fragilidades (arquitecturais e/ou funcionais).

ii. Quantificar o número de instituições portuguesas que actualmente utilizam SL/CA, nomeadamente, em ambiente SIG.

iii. Construir um repositório de organizações nacionais que procederam a um processo de implementação de SL/CA SIG, tentando escrutinar se se trata de casos de sucesso ou de insucesso.

iv. Elencar (se os dados disponíveis o permitirem) lições, boas práticas, vicissitudes e outros tópicos assentes em dados empíricos, tendo em vista a construção de uma matriz de suporte à formulação do MDS4OSS com enfoque nas TIG/SIG.

4. **Jangada de SIG 2.0 – quantificação de custos e a urgência de uma abordagem multifacetada para disponibilização de um guião/manual (plataforma) para suporte à decisão de migração na Administração Pública Portuguesa**

   i. Quantificar os custos relacionados com a aquisição/implementação de SP e SL/CA, por ano, por sector de actividade da APP e por município.

   ii. Aferir qual destas soluções implica um maior esforço monetário.

   iii. Destacar a importância de instituições como a AMA – monitorização, controlo, acompanhamento e fiscalização.

I. Organismos, normas, formatos e protocolos, especificações e *standards*

> *"É bastante bem conhecido que os standards implementados no Software de Código Aberto podem reduzir riscos relacionados com o lock-in, melhorar a interoperabilidade, e promover a competição no mercado."*
>
> Jonas Gamalielsson, Björn Lundell, Jonas Feist, Tomas Gustavsson, Fredric Landqvist (2015:30)[78]

Ao iniciarmos a discussão sobre normas, formatos e protocolos, especificações, *standards* e organismos que as produzem e implementam, devemos, em primeiro lugar, clarificar estes conceitos. Normas são regras que "alguma entidade competente" define e outros se predispõem a cumprir, no caso do SL/CA falamos da OGC e, no caso da cartografia destacamos a importância da ISO. Em tecnologia, nomeadamente na produção de *software*, os desenvolvedores devem obedecer a um conjunto de normas/regras/convenções reunidas num documento chamado "especificação" (guia para desenhar sistemas que usam normas), o seu cumprimento garante que diferentes programas lidem com os mesmos dados, obtendo exactamente o mesmo resultado.

Na mesma linha, as normas estabelecidas para a informação descrevem formas de organizar consistentemente a informação para que ela possa ser entendida e usada por múltiplas aplicações independentes. Entre elas distinguem-se dois tipos de normas: os formatos, usados para guardar informação e, os protocolos, usados para transmitir dados.

Assim, por 'norma aberta' entende-se a norma técnica destinada à publicação, transmissão e armazenamento de informação em suporte digital que cumpra cumulativamente os seguintes requisitos (Pereira, 2012):

i. A norma seja adoptada em resultado de um processo aberto e disponível à participação de todas as partes interessadas;

[78] No original, *"I tis widely acknowledged that standards implemented in open source software can reduce risks for lock-in, improve interoperability, and promote competition on the market*." *In* GAMALIELSON, Jonas; LUNDELL, Björn; FEIST, Jonas; GUSTAVSSON, Tomas & LANDQVIST, Fredric (2015) – "On organisational influences in *software* standards and their open source implementations". **Information and *Software* Technology**, N.º 67, 30-43.

ii. O respectivo documento de especificações tenha sido publicado e livremente disponibilizado, sendo permitida a sua cópia, distribuição e utilização. O mesmo não pode incidir sobre acções e processos não documentados;

iii. Os direitos de propriedade intelectual que lhe sejam aplicáveis, incluindo patentes, tenham sido disponibilizados de forma integral, irrevogável e irreversível ao Estado Português;

iv. Não existam restrições à sua implementação.

Face ao exposto, estas normas, às quais que se contrapõem as normas conhecidas e implementadas apenas por desenvolvedores de SP (fecham o modo como se guarda e se acede à informação – formatos fechados), são consideradas uma ferramenta da maior utilidade no sentido da interoperabilidade, na medida em que permitem alcançar a capacidade de dois ou mais sistemas interagirem e trocarem dados de acordo com um método definido de forma a obter os resultados esperados (Pereira, 2012).

Assim, no âmbito dos SIG e das TIG, qualquer discussão sobre normas e formatos *standard* que implementam, remete inexoravelmente para questões de foro técnico relacionadas com a disponibilização, acesso e análise de informação georreferenciada, por parte de decisores, agentes do mercado económico e cidadãos (Borzacchiello & Craglia, 2013). Por outras palavras, a definição de normas padrão tem a sua razão na necessidade de difundir conhecimento espacial tendo em vista uma tomada de decisão célere convenientemente ajustada à realidade (Vanderhaegen & Muro, 2005; Directiva 2007/2/CE).

As designadas Infra-Estruturas de Dados Espaciais (IDE), sistemas que visam integrar e disponibilizar dados espaciais produzidos por distintas instituições, que geralmente se encontram dispersos e inacessíveis (Maranhão, 2013), surgiram também na perspectiva de sistematizar parâmetros que, cumpridas as normas, permitem organizar de modo adequado e formal a informação geoespacial. Para ser viável, a sua arquitectura pressupõe a interacção entre vários servidores de dados, a catalogação desses dados e serviços de pesquisa, visualização e gestão de *datasets* espácio-temporais (Granell *et al*., 2014). Consequentemente, isto requer sistemas locais que possibilitem a manipulação (interoperável) de dados ou informação por parte de outros agentes locais, o que só se consegue se todos eles estabelecerem políticas de normalização e de harmonização dos dados segundo normas previamente definidas.

Posto isto, qualquer IDE deve colocar em prática a noção de interoperabilidade, bem como, a de transmissão de conhecimentos geográficos através da disponibilização generalizada (para o cidadão

comum e agentes de decisão) de uma apreciável miríade de dados espaciais. Tal como detalharemos mais à frente, a AP Europeia tem vindo a promover a interoperabilidade entre serviços (*eEurope* 2005; Lei n.º 36/2011 de 21 de Junho; Resolução do Conselho de Ministros n.º 91/2012). No âmbito dos SIG, ordenamento do território e política ambiental, isto tem sido feito através da constituição de plataformas (IDE), em primeiro lugar, a uma escala nacional (SNIG[79] e iGEO[80] – caso português) e, em segundo lugar, a uma escala internacional (INSPIRE – âmbito europeu), retirando vários benefícios. De entre eles, destacamos a melhoria da qualidade das políticas sobre o território ou o potenciar do desenvolvimento económico com o incentivo do uso de informação pública em fins comerciais, promovendo-se, assim, a criação de serviços de valor acrescentado, produtos e novos trabalhos, em consonância com a filosofia *open data, open code, open minds*[81] (Vanderhaegen & Muro, 2005; Borzacchiello & Craglia, 2013).

Percebe-se que tudo se torna mais fácil se todos falarem a mesma linguagem, portanto, a opção mais sensata passa pela implementação de normas internacionais disponíveis sobre esta matéria, em específico, as propostas pela ISO.

(Ins)Constituído em 1994, o Comité Técnico 211 da ISO (ISO/TC211) tem desenvolvido uma série de normas internacionais para informação geográfica. Referimo-nos às normas da família ISO19100, que fornece directrizes sobre qualidade cartográfica, modelação de dados geoespaciais e sua gestão, codificação e serviços relacionados, especificando também métodos, ferramentas e serviços para gestão de informações, desde a definição, aquisição, análise, acesso, apresentação e transferência de tais dados em formato digital entre diferentes utilizadores, sistemas e locais (Maranhão, 2013).

Entre elas, as normas de implementação (Tabela XI)[82] incluem em larga medida o trabalho realizado pelo OGC, de tal modo que não será errado caracterizá-las como normas abertas (Matos, 2008). De acordo com o mesmo autor, a actividade da OGC, embora privilegiando a implementação, assenta num profundo estudo teórico e actualmente está em grande consonância com o trabalho desenvolvido na ISO/TC211, cabendo-lhe a produção de normas de implementação que posteriormente são incorporadas como normas ISO 19100. Por exemplo, a ISO 19128, 19136 e 19142 são normas que regem a moldura formal dos protocolos e formatos (*standard*) tutelados pela

[79] http://snig.dgterritorio.pt/portal/ (acedido em 26 Agosto, 2015).
[80] http://www.igeo.pt/ (acedido em 26 Agosto, 2015).
[81] Veja-se, por exemplo, a *Open Data Strategy for Europe* proposta pela Comissão Europeia (http://europa.eu/rapid/press-release_IP-11-1524_en.htm?locale=en., acedido em 26 Agosto, 2015).
[82] De acordo com Matos (2008), as normas ISO 19100 são habitualmente agrupadas em conjuntos segundo a sua natureza: normas singulares; normas de infra-estruturas, normas básicas, normas de imagem, normas de catálogo e normas de implementação.

OGC: *Web Map Service* (WMS), *Web Feature Service* (WFS) e a *Geography Markup Language* (GML) (http://www.opengeospatial.org/standards/is, acedido em 27 Agosto, 2015). A cooperação entre a OGC e a ISO acresce credibilidade às normas abertas e é indiciador (e garante fiel) da sua qualidade.

Tabela XI - Normas de implementação da ISO no âmbito da informação geográfica. Fonte: Matos (2008), p. 260

Item	Título	Documento
19125-1	*Geographic information – Simple feature access – Part I: Common architecture*	ISO 19125-1:2004
19125-2	*Geographic information – Simple feature access – Parte II: SQL option*	ISO 19125-2:2004
19128	*Geographic information – Web Map Server interface*	ISO 19128:2005
19131	*Geographic information – Data product specifications*	N 2058
19132	*Geographic information – Location Based Services – Reference model*	ISO/DIS 19132
19133	*Geographic information – Location Based Services – Tracking and navigation*	ISO 19133:2005
19134	*Geographic information – Location Based Services – Multimodal routing and navigation*	N 2045
19136	*Geographic information – Geography Markup Language*	ISO/DIS 19136
19137	*Geographic information – Core profile of the spatial schema*	ISO/DIS 19137
19138	*Geographic information – Data quality measures*	N 2029 (DTS)
19139	*Geographic information – Metadata – XML schema implementation*	N 2029
19140	*Geographic information amendment process*	
19141	*Geographic information – Schema for moving features*	N 1970
19142	*Geographic information – Web Feature Service*	N 1765
19143	*Geographic information – Filter encoding*	N 1766
19144-1	*Geographic information – Classification Systems – Part 1: Classification system structure*	N 1935
19144-2	*Geographic information – Classification Systems – Part 2: Land Cover Classification System LCCS*	N 1935
19145	*Geographic information – Registry of representations of geographic point locations*	N 1942

Em termos formais, a OGC (Figura 15) resultou do esforço de um conjunto de organismos públicos, universidades e empresas privadas (entre as quais o representante de maior renome internacional em termos de *software* SIG proprietário), e é uma organização concentrada em criar e lançar especificações abertas (não proprietárias) para ultrapassar barreiras que comprometem a implementação na prática do conceito "interoperabilidade" entre programas e sistemas informáticos, promovendo a independência relativamente a desenvolvedores de formatos proprietários; este organismo é composto por inúmeros profissionais de múltiplos campos e produz *standards* consistentes e de alta qualidade (Michael & Ames, 2007). De facto, a OGC e a ISO (Figura 15) trabalham em conjunto para garantir a qualidade dos formatos e protocolos

abertos, propondo ou recolhendo medidas (propostas pela comunidade) que os avaliem e garantam (Sanderson *et al.*, 2007).

Figura 15 - *Open Geospatial Consortium* (OGC) – em cima, e *International Organization for Standardization* (ISO) – em baixo. Fonte: http://www.opengeospatial.org/ogc/vision (acedido em 27 Agosto, 2015); http://qmlsbd.com/international-standards (acedido em 23 Dezembro, 2015).

E, por falar em protocolos que, como antes se referiu, são serviços interoperáveis que garantem a transmissão e o livre acesso a dados geoespaciais que obedecem às normas e formatos *standard*, não podemos deixar de destacar algumas das suas limitações e/ou constrangimentos apontados por autores como Rautenbach *et al.* (2013). A saber:

i. Esforço adicional requerido para programar o serviço WPS (*Web Processing Service*), de modo a que este seja capaz de normalizar e classificar dados, e programar as regras associadas aos estilos SLD (*Styled Layer Descriptor*), *standard* da OGC;

ii. Facto de o formato SLD não ser interoperável com o *ThematicWS*. Assim, segundo os mesmos autores, produzir mapas temáticos com os serviços web da OGC tem como vantagens a interoperabilidade com outros formatos *standard*, a reutilização (adaptação do código) dos serviços e consequente redução de custos. Por seu turno, a principal desvantagem prende-se com a falta de flexibilidade, uma vez que, como referem na sua expressão nativa, "…*with a standard one only gets the standard functionality, nothing more and nothing less* (op. cit. Rautenbach *et al.*, 2013), isto é, em contrapartida, os utilizadores destes protocolos e formatos perdem funcionalidades que lhes podem ser úteis, disponíveis noutros formatos.

iii. Os mesmos autores concluíram, ainda, que persistem muitos obstáculos na criação de infra-estruturas de informação espacial, consequência da falta de serviços normalizados. Apesar de todos os recentes desenvolvimentos que se têm feito na constituição de IDE normalizadas que permitam a interoperabilidade, esta ainda não é totalmente abrangente e estrutural, pois restam ainda muitos organismos/instituições/departamentos/secções/serviços, projectos, etc., que se encontram, ainda, a proceder à conversão dos seus dados para formatos interoperáveis.

Sintetizando, a referência à OGC e à ISO serve, sobretudo para apontar os benefícios e as vicissitudes da adopção de *standards,* mostrando que a produção de normas abertas não é um processo desorganizado e arbitrário entregue a cidadãos anónimos cujas competências são desconhecidas, pelo contrário, é levado a cabo por órgãos credíveis, com elevada reputação no que diz respeito à definição de padrões de qualidade. Independentemente dos problemas referidos, regra geral, a qualidade dos formatos e protocolos da OGC é elevada (Rafoss *et al.*, 2010; Castronova *et al.*, 2013; Díaz *et al.*, 2013; Rautenbach *et al.*, 2013; Chen *et al.*, 2015), o que assegura a consistência lógica dos dados e dos serviços de dados, como comprova o facto de o WFS e WMS terem sido formalmente definidos como *standard* na Resolução do Conselho de Ministros 91/2012 que implementa o RNID.

O SL/CA surge aqui como o modelo de *software* que trabalha exclusivamente com estes formatos e protocolos. Claro que, por si só, tal não assegura a qualidade do *software* mas, uma vez assegurada a interoperabilidade entre formatos e serviços de dados, garante aos utilizadores a liberdade de trabalhar em SIG independentemente do *software* que se utilize, obedecendo ao princípio de que o "raciocínio SIG é independente, e deve prevalecer, sobre o SIG como *software*".

E por falar em qualidade, impõe-se referir outros organismos cujo assinatura de competências é incontornável no que à certificação de SL/CA diz respeito. No que concerne aos SIG/TIG, a OSGeo (Figura 16) é uma associação sem fins lucrativos que tem como objectivo suportar e promover o desenvolvimento e utilização de tecnologias, padrões, e dados livres. Tratando-se de uma fundação, presta também apoio financeiro, organizacional e legal à comunidade de utilizadores de SL/CA de índole geoespacial, e funciona como entidade independente para a qual os membros da comunidade podem contribuir com código, financiamento e quaisquer outros recursos, com o intuito de os rentabilizar em benefício do interesse público (http://www.osgeo.org/content/foundation/about.html, acedido em 27 Agosto, 2015)[83].

[83] Em 2010 foi criada a OSGeo-PT que reinterpreta e implementa os objectivos da OSGeo Internacional em Portugal (http://osgeopt.pt/, acedido em 27 Agosto, 2015).

Figura 16 - *The Open Source Geospatial Foundation* (OSGeo). Fonte: http://www.osgeo.org/content/foundation/about.html (acedido em 27 Agosto, 2015).

Para além da promoção da utilização de tecnologias geoespaciais livres e/ou de código aberto e das várias iniciativas e programas que incrementa, a OSGeo tem como uma das principais metas assegurar que os projectos mantidos na fundação assegurem um elevado grau de qualidade, tendo em vista a construção e preservação da sua 'marca'.

A Figura 17 dá-nos conta de um inventário de vários projectos que, até esta data, fazem parte integrante da fundação. Destaca-se a maior representatividade de projectos relacionados com o mapeamento *web*; por seu turno, as aplicações *desktop*, apesar da sua grande diversidade e variedade (cfr. Tabela IX), são apenas três, aquelas que, com excepção do Marble[84], são frequentemente reconhecidas pelo seu maior índice de utilização e fiabilidade, sobretudo o *Geographic Resources Analysis Support System* (GRASS). Esta evidência significa uma de duas coisas mutuamente exclusíveis:

Figura 17 - Projectos mantidos pela OSGeo. Fonte: http://www.osgeo.org/ (acedido em 27 Agosto, 2015).

[84] O Marble, segundo o nosso ponto de vista, não é uma verdadeira aplicação SIG *desktop*, pois permite somente a visualização de dados e informação geográfica. É, de algum modo, uma versão *open source*, muito mais limitada, parente próxima do Google Earth.

i. Ou os restantes projectos encontram-se ainda numa fase de desenvolvimento incipiente, e, por isso, não respeitam os padrões de qualidade estabelecidos pela organização;

ii. Ou, em alternativa, não têm interesse em pertencer à fundação, hipótese menos provável, pois são conhecidas as dificuldades que os projectos têm em subsistir, nomeadamente, por motivos de financiamento.

Na realidade, a OSGeo deixa transparecer uma atitude altamente selectiva no que diz respeito aos projectos que a integram. Numa primeira fase, qualquer projecto que pretenda fazer parte da organização tem de ser incubado. A incubadora da OSGeo é metaforicamente um inquisidor, uma vez que avalia os projectos segundo um conjunto de princípios e critérios que devem ser escrupulosamente cumpridos pela equipa do projecto. Este processo não é só um momento de escrutínio, pois dele resultam constatações sobre os aspectos que se devem melhorar; assim, a colaboração entre a OSGeo e os gestores de determinado projecto traduz-se numa melhoria gradual das propriedades (intrínsecas e extrínsecas) do *software* até ao momento em que este passa a ser um projecto OSGeo. Actualmente, os projectos elencados na Figura 18 encontram-se em fase de incubação, só depois de confirmadas todas as condições definidas pela OSGeo é que estes podem envergar a sua bandeira.

Figura 18 - Projectos incubados pela OSGeo. Fonte: http://www.osgeo.org/ (acedido em 27 Agosto, 2015).

Acresce, ainda, que a OSGeo acompanha todos os projectos, publicando relatórios sobre o seu estado, evoluções e aspectos que ainda necessitam de melhoria (https://journal.osgeo.org/index.php/journal, acedido em 27 Agosto, 2015).

Em suma, a acção da OSGeo traduz-se numa melhoria gradual da tecnologia, por um lado, pela intervenção directa sobre o *software* e, por outro, pela constituição de uma comunidade ampla de utilizadores que partilham experiências e promovem a entreajuda. A intersecção desta acção com as normas propostas pela ISO e OGC fazem do SL/CA SIG e dos formatos livres soluções confiáveis e altamente interoperáveis, sendo cada vez mais visíveis os benefícios da sua utilização conjunta.

2. O panorama legislativo português e a obrigatoriedade de implementação de *software* de baixo custo

> *"As despesas com aquisição de licenças de software, …, apenas poderão ser executadas nos casos em que seja fundamentadamente demonstrada a inexistência de soluções alternativas com software livre ou que o custo total de utilização da solução em software livre seja superior à solução em software proprietário ou sujeito a licenciamento específico, incluindo nestes todos os eventuais custos de manutenção, adaptação, migração ou saída."*
>
> OE (2013)[85]

A legislação que intima os órgãos da APP a justificarem as suas opções em matéria de escolha e aquisição de *software* é, para nós, o factor motivacional mais relevante para a realização desta dissertação, aquele que nos leva a tentar provar que o SL/CA SIG é uma alternativa credível e viável. Face ao exposto, tornava-se incontornável a necessidade de inventariar e dar conta (Figura 19) do apreciável lote de documentos que conduziram a esta obrigatoriedade, de modo a perceber concretamente o que estes textos tentam incrementar na realidade (inspiração; razões; órgãos alvo; condições; excepções e limitações).

A nossa análise inicia-se no ano de 2002, aquando da promulgação da Resolução do Conselho de Ministros n.º 21/2002. Esta Resolução transpõe para o âmbito nacional a letra da iniciativa *eEurope* 2002, um plano de acção que visava estimular serviços, aplicações e conteúdos seguros assentes numa infra-estrutura de banda larga amplamente disponível, fomentando a utilização de programas de código aberto no sector público. Seguiu-se o *eEurope* 2005, que visava reforçar a interoperabilidade[86] entre os vários organismos públicos dos Estados membros, tendo em vista duas metras que consideramos nucleares:

[85] Lei n.º 66-B/2012 de 31 de Dezembro (Orçamento do Estado para 2013). Diário da República, 1.ª Série – N.º 252 – de 31 de Dezembro de 2012. Lisboa. Portugal.

[86] Capacidade de dois ou mais sistemas, designadamente computadores, meios de comunicação, redes, *software* e outros componentes de tecnologia da informação, de interagir e de trocar dados de acordo com um método definido de forma a obter os resultados esperados (Lei n.º 36/2011 de 21 de Junho).

i. Ligação das administrações públicas, escolas e cuidados de saúde em banda larga;

ii. Serviços públicos interactivos, acessíveis para todos e oferecidos em múltiplas plataformas.

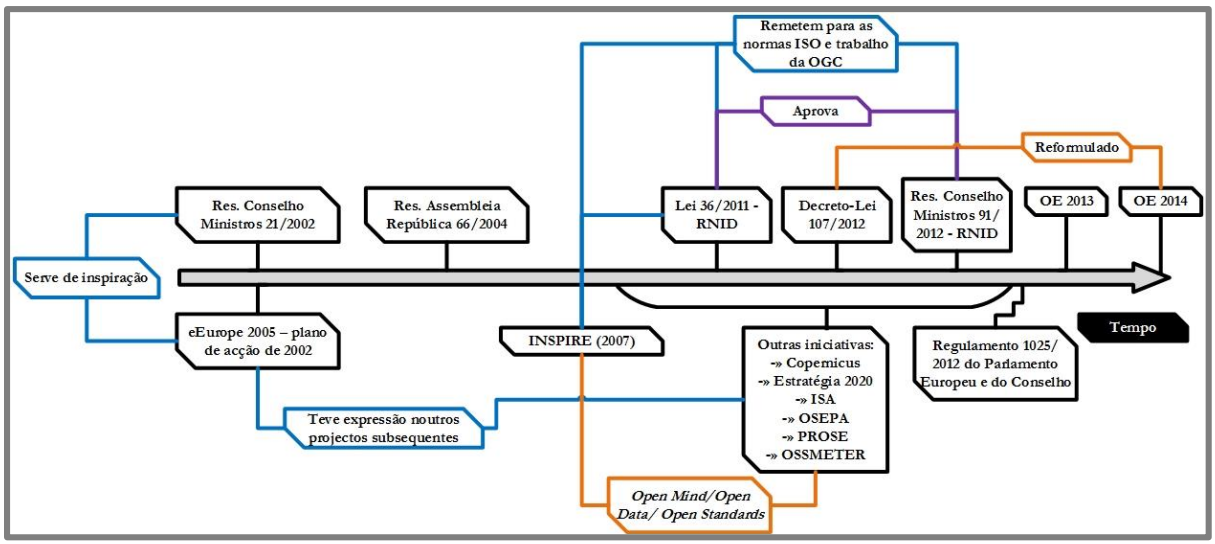


Figura 19 - Fundamentação formal dos diplomas que conduziram à recente obrigatoriedade de justificação das opções em matéria de selecção de *software* de acordo com alguns dos principais dispositivos e manados de regulamentação (europeia e nacional).

O *eEurope* não é propriamente um enunciado de incentivo à utilização de SL/CA (porque se foca em questões relacionadas com interoperabilidade); não obstante, não deixa de ser expressa a *intenção de apoiar a normalização com vista a uma maior utilização de normas abertas e de software de fonte aberta* (op. cit. *eEurope* 2005, p. 18). Neste ponto, a UE já tem como objectivo a implementação de normas abertas que fomentem a interoperabilidade; claro, a consequência lógica é promover-se também o SL/CA, por ser aquele que trabalha exclusivamente com normas abertas[87].

Assim, a Resolução 21/2002 é reflexo dos objectivos da acção *eEurope* 2002, que transpõe para a lei portuguesa, nomeadamente no caso da promoção da utilização de sistemas abertos de *software* pela APP.

[87] Norma técnica destinada à publicação, transmissão e armazenamento de informação em suporte digital que cumpra cumulativamente os seguintes requisitos: a sua adopção decorra de um processo de decisão aberto e disponível à participação de todas as partes interessadas; o respectivo documento de especificações tenha sido publicado e livremente disponibilizado, sendo permitida a sua cópia, distribuição e utilização, sem restrições; o respectivo documento de especificações não incida sobre acções ou processos não documentados; os direitos de propriedade intelectual que lhe sejam aplicáveis, incluindo patentes, tenham sido disponibilizados de forma integral, irrevogável e irreversível ao Estado Português; não existam restrições à sua implementação (Lei n.º 36/2011 de 21 de Junho).

De notar que em ambos os documentos é referida a importância da partilha de boas práticas, a partir das quais se podem agilizar processos de migração para o sistema aberto interoperável. Tal preocupação perdurou e está bem presente na Resolução da Assembleia da República 66/2004, que sublinha a necessidade de criação de serviços de partilha de informação sobre experiências com SL/CA, tendo em vista a definição de cenários e linhas de intervenção; na mesma linha, promove também a criação de um serviço de apoio para suporte técnico à implementação e optimização de soluções abertas, de *software*.

Desta política surgiram, desde então, algumas plataformas *web* em que está presente este espírito de partilha empírica (repositórios de dados, informações e conhecimentos sobre a adopção de SL/CA, nomeadamente na APP), destacando-se os exemplos do *Software* Publico.gov.pt (beta) e da *Software* Livre@AP[88]. Para além destes, na mesma linha, destaca-se também a plataforma "Interoperabilidade na Administração Pública" (IAP) da responsabilidade da AMA, que visa dotar a APP de ferramentas partilhadas para a interligação de sistemas, federação de identidades, fornecedor de autenticação, *messaging*, pagamentos, entre outras, que permitam, de uma forma ágil e com economia de escala, a composição e disponibilização de serviços electrónicos multicanal mais próximos das necessidades do cidadão e das empresas (http://www.iap.gov.pt/Default.aspx, acedido em 29 Setembro, 2015).

Os sistemas abertos têm sido uma das mais importantes cruzadas da UE, expressa noutros programas subsequentes à *eEurope*, tais como a Directiva INSPIRE, o programa Copernicus, o projecto OSEPA, PROSE, OSSMETER, ou, ISA. Como o SL/CA não é dissociável dos *open standards*, em cada uma destas iniciativas fomenta-se a utilização deste tipo de recurso informático na AP europeia. A mentalidade *open data, open mind, open source* está, como já antes se disse, cada vez mais presente na política europeia, e é também uma ideia bem vincada na Estratégia Europa 2020.

Em Portugal, na sequência dos documentos referidos, a adopção de normas abertas materializa-se verdadeiramente com a sua promulgação em Diário da República, tendo em vista a promoção da liberdade dos cidadãos e organizações e da interoperabilidade dos sistemas informáticos do Estado (Lei n.º 36/2011 de 21 de Junho). A referida Lei adianta as disposições gerais do RNID (que deverá

[88] http://www.*software*publico.gov.pt; http://www.*software*livre.gov.pt/, respectivamente (acedido em 29 Março, 2015). A primeira plataforma é uma iniciativa da Rede Comum do Conhecimento patrocinada pela AMA. Caracteriza-se por ser um canal de partilha de boas práticas, notícias, eventos e documentação relacionada com aquilo que se define como *software* público. A segunda é um repositório de conhecimento sobre SL/CA que as entidades do Estado Português detêm, destinando-se a ser um ponto de encontro e de troca de experiências entre todos aqueles que, ao serviço do Estado, o utilizam. Surge no âmbito dos trabalhos de coordenação da utilização de TI na APP que são assegurados pela Comissão Intersectorial de Tecnologias de Informação para a Administração Pública, e que tem como entidades promotoras o Instituto Nacional de Estatística, o Centro de Informática do Exército, a Secretaria Geral do Ministério da Educação, a Secretaria Geral do Ministério da Cultura e o Instituto de Informática.

ser elaborado pela AMA), no qual os SIG (incluindo cartografia, cadastro digital, topografia e modelação) têm de ser necessariamente contemplados, remetendo inevitavelmente para o trabalho da OGC (que estipula normas *standard* e regras de interoperabilidade).

Mais tarde, a Resolução do Conselho de Ministros n.º 91/2012 aprova e estabelece o RNID, o qual deve ser aplicado, nos termos previstos no artigo 4º da Lei n.º 36/2011, a todos os processos de implementação, licenciamento ou evolução de sistemas informáticos na Administração Pública, salientando que todos os documentos de texto em formato digital que sejam objecto de emissão, intercâmbio, arquivo e ou publicação pela Administração Pública. Os SIG são contemplados apenas com a definição dos protocolos web da OGC (WMS e WFS) como *standards*, deixando-se de lado os formatos que possibilitam o armazenamento de informação na memória interna do computador.

Até ao momento, referiram-se os diplomas que, implicitamente, promovem a adopção de SL/CA pela sua estrita relação com as normas abertas e consequentemente com a interoperabilidade. Este ponto é, desde logo, suficiente para justificar um processo de migração, justificação que se vê reforçada, ainda em 2012, na justa medida em que surgem outros diplomas que levam impreterivelmente as instituições da APP a olhar a migração para SL/CA como um bom exemplo de redução da despesa.

A Resolução do Conselho de Ministros 12/2012 menciona a Resolução do Conselho de Ministros 46/2011 que constituiu o Grupo de Projecto para as Tecnologias de Informação e Comunicação (GPTIC) que elaborou o Plano Global Estratégico de Racionalização e Redução dos Custos nas TIC na Administração Pública (PGERRCTIC). Com efeito, segundo os termos do diploma de 2012, a implementação de uma estratégia global para redução das despesas na APP exige o cumprimento das orientações deste plano estratégico, cujo objectivo final é conseguir alcançar ganhos de poupança e de eficiência para a máquina administrativa ministerial, e cujos resultados são de, algum modo, expectáveis e garantidos.

Nestes termos, o PGERRCTIC estabelece 25 medidas transversais de racionalização das TIC (com impacto em toda a APP), proporcionando uma redução (até 500 milhões de euros) na despesa de funcionamento anual da Administração Central. A Resolução 12/2012 implementa estas 25 medidas, entre as quais há uma que remete para a exigência da avaliação de projectos e despesas TIC. Segundo o dito plano, deve-se *implementar um processo de avaliação de projectos e despesas da TIC obrigatório e vinculativo, estabelecendo mecanismos formais de avaliação multicritério dos investimentos, garantindo que apenas são financiados e implementados os projectos que demonstrem reais garantias de retorno nas várias*

dimensões em análise, minimizando investimentos redundantes e desalinhados com as políticas nacionais para as TIC na APP.

Como corolário natural, é fácil percebermos que isto implica a definição de uma metodologia de avaliação e de um modelo organizacional de suporte à aplicação da metodologia, quer a nível global quer a nível sectorial. Nesta linha, o Decreto-Lei 107/2012 vem dar cumprimento à medida apresentada, criando um processo de avaliação prévia, obrigatório e vinculativo, dos investimentos especialmente relevantes com a aquisição de bens e serviços com as TIC, e cuja responsabilidade pela recepção, análise, avaliação, acompanhamento e monitorização deste processo é da competência da AMA. Não obstante, são contempladas algumas excepções; os organismos e/ou tipos de contratos que se encontrem numa das condições seguintes não são alvo do pedido de parecer:

i. Contratações cujo contrato seja declarado secreto ou que a respectiva execução exija medidas especiais de segurança;

ii. Contratações cujo adjudicante seja um serviço da administração indirecta ou uma entidade do sector empresarial do Estado (empresas cuja principal função é a produção de bens e serviços mercantis e financeiros, e que por isso se encontrem em regime de concorrência no mercado);

iii. Contratações que têm em vista a aquisição, manutenção ou evolução de sistemas operacionais críticos;

iv. O decreto não é aplicável às entidades administrativas independentes (Comissão Nacional de Eleições, Entidade Reguladora para a Comunicação Social e Comissão de Mercado de Valores Mobiliários), ao Banco de Portugal e aos estabelecimentos de ensino superior.

v. O decreto aplica-se apenas às aquisições e prestações de serviços que tenham valor igual ou superior a 10 mil euros, devendo o estudo ser aplicado ao SL/CA bem como às soluções proprietárias.

Os regimes excepcionais são relevantes e constituem limitações desta iniciativa. Destacamos duas que nos parecem ser essenciais:

i. A exclusão de uma fatia considerável dos organismos públicos, nomeadamente instituições de ensino superior, compromete a redução dos exageros tidos com a aquisição de SP. É no ambiente académico que o SL/CA faz mais sentido, pois pode

ser estudado e melhorado, isto é, a análise do código-fonte é conversível numa estratégia pedagógica em benefício dos discentes.

ii. Sendo possível adquirir *software* ou serviços abaixo dos 10 mil euros sem pedido de parecer, torna-se relativamente fácil contornar esta obrigatoriedade (um contrato de 15 mil euros pode ser transformado em dois de 7500 euros, convertendo-se numa modalidade bi ou plurianual, no caso de estarmos perante valores mais avultados).

Para além disto, se o organismo/instituição fizer um pedido de parecer à AMA e se este último não for emitido no prazo de 30 dias, a situação transita de forma automática, mas legal, para uma situação que equivale à emissão de parecer positivo; ou seja, num momento em que a AMA não consegue emitir os pareceres em tempo legal, por exemplo, por não ter meios suficientes para processar um número elevado de solicitações, tal implica que um organismo/instituição da APP possa adquirir *software* sem quaisquer restrições, seja ele proprietário ou livre.

Fazemos notar que, até ao momento, focámos esta abordagem na questão da redução da despesa e da sua legitimação pela via dos dispositivos regulamentares, ou seja, não há uma referência explícita ao SL/CA enquanto agente que se integra nessa política nem tampouco enquanto opção cujo valor, enquanto tal, esteja inequivocamente demonstrado. Percebendo que o SL/CA tem um importante papel no desiderato de equilibrar a despesa pública, o OE de 2013 define claramente que as despesas com aquisição de licenças de *software* (previstas nos vários orçamentos) só poderão ser executadas nos casos em que seja fundamentadamente demonstrada a inexistência de soluções alternativas em SL/CA ou que o seu custo total exceda o da solução em SP. A letra deste orçamento vem reforçar o disposto no Decreto 107/2012, acrescentando-lhe a obrigatoriedade de conceder preferência à utilização de SL/CA.

Na sua sequência, também o OE de 2014 introduz algumas alterações ao Decreto 107/2012, ainda que mais formais e com poucas implicações para a substância do documento. Por exemplo, adiciona-se-lhe a definição de SL, e delimitam-se os aspectos a serem considerados no custo total de utilização da solução, nomeadamente o licenciamento, manutenção, adaptação e desenvolvimento, migração, saída e formação.

Enfim, os OE de 2013 e de 2014 vinculam, à questão da análise do custo total da solução a ser implementada, a preferência que se deve conceder ao SL/CA, ou seja, estes acrescentam ao decreto que lhes precedeu referências implícitas de que a avaliação metodológica comparada deve contrapor soluções proprietárias e livres e/ou de código aberto.

Actualmente, aguardamos com expectativa a revisão do RNID (este documento deve ser revisto num prazo máximo de três anos), que se encontra em curso. Esta revisão trará para a legislação portuguesa as considerações que constam do Regulamento 1025/2012 do Parlamento Europeu e Conselho, o último diploma europeu relevante em matéria de normas europeias. Segundo este regulamento, as normas adoptadas por parte dos Estados-Membros devem ser suportadas em normas europeias e aprovadas por uma organização europeia de normalização (tal como a CEN[89], Cenelec[90] ou a ETSI[91]) em coordenação com organismos internacionais de normalização (ISO).

Em jeito de síntese e de remate conclusivo deste ponto do trabalho, destacamos os dois papéis que o SL/CA representa no panorama legislativo português. Num primeiro momento, com o interesse de se obter uma rede informática interoperável e comunicativa – estatal (escala local) e europeia -, recomendou-se a adopção de normas abertas, garantindo-se a coerência e a legibilidade das informações que circulam entre os vários órgãos públicos e, destes, para os cidadãos. Pela sua consonância com os *open standards*, o SL/CA é, sem dúvida, a tecnologia de eleição para servir de suporte à máquina que interconecta toda a Administração Pública.

Num segundo momento, perante a necessidade de redução da despesa, o SL/CA surge como a solução com maior potencial para ser mais rentável. Assim, a APP vê-se actualmente forçada a proceder a uma avaliação rigorosa dos custos de diferentes soluções informáticas (SP e SL/CA) com capacidade para ser o suporte das suas actividades e regular funcionamento.

Este é o ponto que importa reter, pois, mesmo não sendo o único factor que nos ajuda a responder à questão "porque se deve considerar SL/CA na APP?", é aquele que atribui uma grande margem de legitimidade aos objectivos desta dissertação. Inclusive, se pensarmos que estes problemas não estão descontextualizados dos vários esforços da UE, plasmados em missivas emanadas por planos/programas/directivas/entidades como as já referidas (OSEPA, OSSMETER, ISA), mas também não escapam ao empenho da comunidade científica em estabelecer um modelo operacional que satisfaça a recomendação legislativa exposta anteriormente, somos conduzidos a recordar que o presente documento tem por meta, a disponibilização de um documento de apoio, não só para entidades portuguesas, mas também internacionais (*mutatis mutandis*), na busca de uma solução para este problema.

[89] *European Committee for Standardization* - https://www.cen.eu/Pages/default.aspx (acedido em 24 Novembro, 2015).
[90] http://www.cencenelec.eu/Pages/default.aspx (acedido em 24 Novembro, 2015).
[91] *European Telecommunications Standards Institute* - http://www.etsi.org/ (acedido em 24 Novembro, 2015).

3. O *statu quo* da adopção de *Software* Livre e de Código Aberto SIG, no mundo e em Portugal

> *"Abordar um problema que também o é em outras organizações pode aumentar significativamente o rácio de recursos de know-how disponíveis sobre como o resolver – problemas partilhados oferecem soluções partilhadas"*
>
> OSEPA (2012) [92]

Para além dos órgãos que certificam e implementam padrões de qualidade no *software* e nos formatos/protocolos, e da avaliação da eficácia/eficiência dos programas na resolução das tarefas, a existência de casos de sucesso é um indicador importante que atesta as capacidades que esta solução tem para resolver problemas específicos[93]. Mais do que isso, os exemplos de boas (ou más) práticas fornecem-nos boas indicações sobre o *software* a que poderemos dar alguma preferência, propondo estratégias de adopção mais robustas e aperfeiçoadas, aprendendo com os erros do passado.

i. No mundo – O que nos é dado a ver a partir dos exemplos vindos de fora

Na nossa óptica, os exemplos de sucesso e de boas práticas, para além de atestarem a qualidade do SL/CA, constituem uma experiência empírica de valor inestimável para quem se encontra num contexto de tomada de decisão no sentido de eleger a solução mais rentável para a sua organização. Porém, segundo a definição que atribuímos ao termo 'informação empírica', no âmbito dos SIG, concluímos que este tipo de informação é praticamente inexistente. Os dados existentes (que lhe servem de suporte) indicam somente a iniciativa de adopção e discutem aspectos genéricos que lhe

[92] No original, "*Addressing a problem that is also acknowledged by other organisations can significantly increase the rate of available know-how resources and support as shared problems often have shared solutions." In* OSEPA (2012) – Good Practice Guide covering various aspects of FOSS usage by European Public Administrations. 81 p. Open Source *Software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

[93] Um caso de sucesso é a expressão usada para nos referirmos a processos e metodologias que, em contexto de migração, se mostraram eficazes num contexto específico (organização, área de actuação, região geográfica) e que, por conseguinte, apresentam evidências de que também podem ser eficazes noutros contextos e situações (OSEPA, 2012a).

estão associados; a juntar a isto, eles referem-se normalmente à implementação de soluções como sistemas operativos, servidores e gestão de redes e ferramentas de produtividade e de escritório, ou seja, são menos frequentes as referências à adopção de SL/CA SIG.

Na mesma linha, grande parte das investigações científicas neste âmbito (aquisição de dados empíricos), denominadas pelos seus autores como *empirical study*, não permitem conotar determinado caso de implementação de SL/CA como um caso de sucesso ou insucesso, muito menos averiguar as razões que poderão suportar esse rótulo. A maior parte dos estudos reveste um conjunto de generalizações que, por norma, são omissos relativamente a parâmetros como:

i. Solução implementada;

ii. Objectivos da organização com a sua adopção;

iii. Características da mesma e impacto da solução no pessoal ao serviço;

iv. Procedimentos (tecnológicos) e tomadas de decisão inerentes à migração (que visaram certamente a resolução de problemas que foram surgindo);

v. Indicadores sobre a qualidade e condicionalismos daquela solução, etc.[94].

Deste modo, entenda-se por 'informações empíricas', a lista (mais ou menos extensa e exaustiva) de procedimentos que conduziram à descrição do processo de migração, tais como, vivências, problemas ocorridos e limitações da estratégia adoptada. Esta descrição terá de ser, necessariamente, o mais exaustiva possível, explicitando de modo claro o contexto da organização – características, objectivos, redes e suporte, estratégias, procedimentos tecnológicos usados, metodologias para ultrapassar quaisquer atritos com a resistência do pessoal ao serviço, etc. Só com estes pormenores é possível reter as devidas lições para aqueles que equacionam a possibilidade de migração.

No que diz respeito ao SL/CA em geral, para além dos autores que incrementam processos e análises (estatísticas ou outras) tendo por base amostras de casos reais de implementação (Dedrick

[94] Por exemplo, é difícil encontrar dados sobre o desempenho de um *software* específico em contexto real e aplicado, indicador imprescindível para quem intenta delinear a transição para SL/CA; o baixo custo pode ser um factor potenciador da adopção, porém, como o mesmo programa informático pode ter custos diferentes em duas organizações, a generalização só pode ser feita sabendo exactamente as características da organização e o que esta pretende alcançar com a solução; só assim uma outra entidade, motivada pelos mesmos objectivos, saberá que a tecnologia "A" ou "B" é a que mais favorece a sua actividade. Referir-se que uma tecnologia (ferramenta) tem um elevado ou reduzido desempenho é insuficiente, pois suscitam-se questões como - qual sistema o operativo em que o *software* correu, que ferramentas se usaram no processo de implementação, etc.; não é, portanto, suficiente dizer que determinada solução concreta tem um suporte pouco robusto, importa especificar que tal sucede, por exemplo, porque as empresas (daquele ramo) são de pequena dimensão e não têm capacidade suficiente para responder a todas as solicitações.

& West, 2004; Glynn *et al.*, 2005; Fernandes, 2010; Qu *et al.*, 2010; Fitzgerald *et al.*, 2011; Ven & Verelst, 2011; Godinho, 2012; Marsan *et al.*, 2012a; OSEPA, 2012c; Spinellis & Giannikas, 2012; Engelhardt & Freytag, 2013; Marsan & Paré, 2013; Sarrab & Rehman, 2014; Gallego *et al.*, 2015)[95], distinguem-se outros documentos ou plataformas que citam vários casos de implementação, como as que seguidamente destacamos:

i. http://www.*software*livre.gov.pt/ (acedidos em 29 Agosto, 2015) - casos de implementação na Europa e em Portugal;

ii. http://www.*software*publico.gov.pt/inicio (acedidos em 29 Agosto, 2015);

iii. http://www.participa.br/*software*publico (acedidos em 29 Agosto, 2015) - Brasil;

iv. http://osepa.eu/ (acedidos em 29 Agosto, 2015);

v. https://joinup.ec.europa.eu/community/osor/home (acedido em 29 Agosto, 2015);

vi. Entre outros.

Como se disse, embora os gestores da maior parte destas plataformas falem em 'casos de boas práticas', a grande maioria dos dados factuais refere-se apenas a casos de adopção, que não nos permitem, com rigor, aferir se se tratam de exemplos de boas práticas ou exemplos de sucesso, pois nem sempre há um esforço de monitorização dos projectos de adopção. Em alguns casos, referem-se somente intenções políticas de adopção sem se apresentarem exemplos concretos. Inclusive, na maior parte das vezes são negligenciadas ou omitidas quaisquer metodologias para conferir o título de "sucesso" ou "insucesso"; os investigadores do OSEPA constituem uma das raras excepções, propondo um conjunto de critérios com o intuito de tornar mais objectivo e transparente todo o processo (Tabela XII).

Tabela XII - Campos e critérios de avaliação das práticas com SL/CA. Fonte: OSEPA (2012c), p. 14

Campos de avaliação	Critério de Avaliação	Objectivos
Impacto da solução	Nível de impacto da solução	Verificar se a implementação de SL/CA responde a uma banda larga de assuntos e problemas partilhados por outras organizações em diferentes regiões e condições
Objectivos alcançados e resultados produzidos	Tipo de objectivos alcançados e resultados produzidos	Identificar o tipo e carácter dos objectivos concretizados e resultados gerados por uma prática específica

[95] Como se imagina, quase nenhum dos autores - Fitzgerald *et al.* (2011) constitui uma das poucas excepções -, identifica as instituições que fazem parte da sua amostra, indicando somente o número total de entidades que a constituem.

Avaliações aplicadas anteriormente	Tipo e nível de metodologias de avaliação aplicadas	Identificar o tipo e o nível de qualquer validação ou avaliação anterior
Problemas encontrados durante a implementação	Extensão dos problemas encontrados durante a implementação	Avaliar a extensão dos problemas encontrados e as principais dificuldades sentidas durante a implementação de uma prática
Desenvolvimento e customização do *software*	Nível de desenvolvimento e de customização do *software*	Perceber o nível de maturidade do programa ou o nível de modificações necessárias para a sua implementação
Uso de recursos próprios	Extensão do uso de recursos próprios	Aferir se a adopção resultou da utilização de funcionários e recursos internos ou foi conseguida por associados externos
Transferibilidade	Nível de transferibilidade	Escrutinar a viabilidade da transferência das boas práticas para outros contextos geográficos e organizacionais

Entre as excepções encontra-se também o projecto *Open Source Observatory*, que, ao referir vários exemplos de sucesso, aponta alguns aspectos que permitem caracterizar o retorno do investimento.

Fitzgerald *et al.* (2011) e Marsan & Paré (2013) também nos oferecem elementos que aludem ao sucesso ou insucesso. Por exemplo, os primeiros destacam a adopção de SL/CA na Extremadura (Espanha) como um caso paradigmático daquilo que é uma adopção de sucesso.

Outro exemplo é a utilização de SL/CA no Departamento da Defesa dos Estados Unidos da América – um relatório que procurava elencar todas as aplicações SL/CA e coleccionar exemplos de como essas aplicações estão a ser usadas, concluiu que o SL/CA (representado por 115 programas) desempenha um papel crítico neste departamento, actuando em quatro áreas essenciais: suporte a infra-estruturas, desenvolvimento de *software*, segurança e investigação (Mitre, 2003). Este é um caso de sucesso relevante, tal como sabemos, um departamento com estas responsabilidades valoriza acima de tudo a segurança, e escolheu SL/CA exactamente por permitir o cumprimento dessa condição. Com efeito, a utilização de diversos SL/CA, ao contrário do que se invoca num dos muitos mitos que a ele são atribuídos, pode até constituir uma medida de segurança, visto que permite evitar uma dependência relativamente a um único fornecedor de *software* (o emprego de apenas uma família de *software* tornaria a pirataria muito mais simples e perigosa, isto porque o código teria a mesma génese). Atente-se que nenhuma das aplicações referidas neste relatório tem algum momento de aplicação na área de actuação das TIG.

Em concreto sobre os SIG, não é tarefa fácil encontrar exemplos de implementação internacionais na *web*. Destacamos os seguintes casos:

i. Em 2006, a cidade Schoten (Bélgica) decidiu criar uma plataforma que respondesse à necessidade de reunião, organização e gestão de toda a informação geográfica do município numa única plataforma, de modo a que todas as entidades municipais da Flandres possam usufruir dessa informação (OSEPA, 2012c). Mediante a avaliação de alguns critérios, os decisores optaram pela utilização do PostGIS e PostgreSQL para armazenamento e gestão de dados; Geoserver para acesso aos dados; e do GIM *WebGIS* para gerir os geoportais disponibilizados ao público. Segundo os investigadores do OSEPA foram alcançadas as seguintes metas:

   a. Melhoria da performance e da eficácia da organização;

   b. Independência relativamente aos fornecedores;

   c. Optimização de processos na organização;

   d. Garantia de segurança dos dados e a interoperabilidade entre sistemas.

ii. O Laboratório de Pesquisa de Engenharia Civil do Exercício dos Estados Unidos da América é o exemplo mais conhecido de utilização de um *software* SIG livre. Este organismo desenvolveu, difundiu e utilizou o GRASS GIS como principal solução SIG até meados da década de 90, altura em que se viu obrigada a migrar para sistemas proprietários, acompanhando a tendência de todo o exército norte-americano, o que teve como principal consequência o abandono do projecto GRASS GIS (Neteler & Mitasova, 2008).

ii. Em Portugal – O que nos é dado a ver a partir de exemplos vindos de dentro

Em Portugal, são conhecidas boas práticas associadas a iniciativas de migração para o uso de SL/CA SIG, protagonizadas por várias instituições públicas, das quais se podem destacar: Escola Secundária Jaime Moniz, Departamento de Geografia da Faculdade de Letras da Universidade de Coimbra (Licenciatura em Geografia e Mestrado em TIG), a Câmara Municipal da Mealhada, Câmara Municipal de Oliveira de Azeméis, Câmara Municipal de Mirandela e Câmara Municipal da Ribeira Grande. De todos estes exemplos, este último é o único que consta da plataforma *Software Publico.gov.pt*[96], numa página em que se descrevem superficialmente os contornos em que se

[96] http://www.*software*publico.gov.pt/boas-praticas/-/asset_publisher/wt25j374QVfb/content/sig-sistema-de-informacao-geografica-da-ribeira-grande?redirect=http%3A%2F%2Fwww.*software*publico.gov.pt%2Fboas-praticas%3Fp_p_id%3D101_INSTANCE_wt25j374QVfb%26p_p_lifecycle%3D0%26p_p_state%3Dnormal%26p_p_mode%3Dview%26p_p_col_id%3Dcolumn-1%26p_p_col_count%3D1%26_101_INSTANCE_wt25j374QVfb_advancedSearch%3Dfalse%26_101_INSTANC

processou a migração, de tal modo que se percebe que os objectivos foram cumpridos, mas não se sabe das dificuldades enfrentadas pelos técnicos.

De facto, é impossível negar a falta de dados empíricos detalhados assim como os definimos. Portanto, com o intuito de ampliar a lista apresentada anteriormente, procuramos outras fontes e métodos para adquirir informações sobre o estado da arte em termos de adopção de SL/CA na APP:

i. Análise dos resultados do Inquérito à Utilização das Tecnologias da Informação e Comunicação na Administração Pública Central, Regional e Câmaras Municipais (IUTIC);

ii. Inferência de prováveis casos de migração a partir da BD que armazena os dados de despesa com aquisição/implementação de *software* na APP;

iii. Elaboração e difusão pública de um inquérito por questionário (Anexo B), recorrendo à plataforma LimeSurvey[97].

3.1. Panorama geral dado pelo Inquérito à Utilização das Tecnologias da Informação e Comunicação na Administração Pública

O IUTIC de 2013 demonstra o estado e a evolução da adopção de SL/CA na APP Central e Local (Câmaras Municipais). No que diz respeito à primeira (Figura 20), considerando o SL/CA para servidores de internet e para sistemas operativos, regista-se uma tendência para o aumento da percentagem de organismos que adoptaram ou migraram as suas soluções, embora para a segunda tipologia tenha havido recuos em 2011 (-1%) e em 2013 (-3%), indiciando o retorno, de alguns organismos, ao SP. Destes dados destaca-se a elevada percentagem de organismos que assentam as suas soluções em tecnologia livre, pois, nos últimos anos, mais de metade da administração central usava este tipo de recurso na gestão dos seus servidores, e quase metade já usa sistemas operativos livres ou de código aberto.

E_wt25j374QVfb_keywords%3D%26_101_INSTANCE_wt25j374QVfb_delta%3D5%26p_r_p_564233524_resetCur%3Dfalse%26_101_INSTANCE_wt25j374QVfb_cur%3D3%26_101_INSTANCE_wt25j374QVfb_andOperator%3Dtrue (acedido em 27 Novembro, 2015).

[97] A difusão deste inquérito através da plataforma LimeSurvey (integrada no sistema informática do Centro de Informação da Universidade de Coimbra) só foi possível com a colaboração do Engenheiro João Sá Marta, a quem agradecemos pela pronta resposta ao nosso pedido.

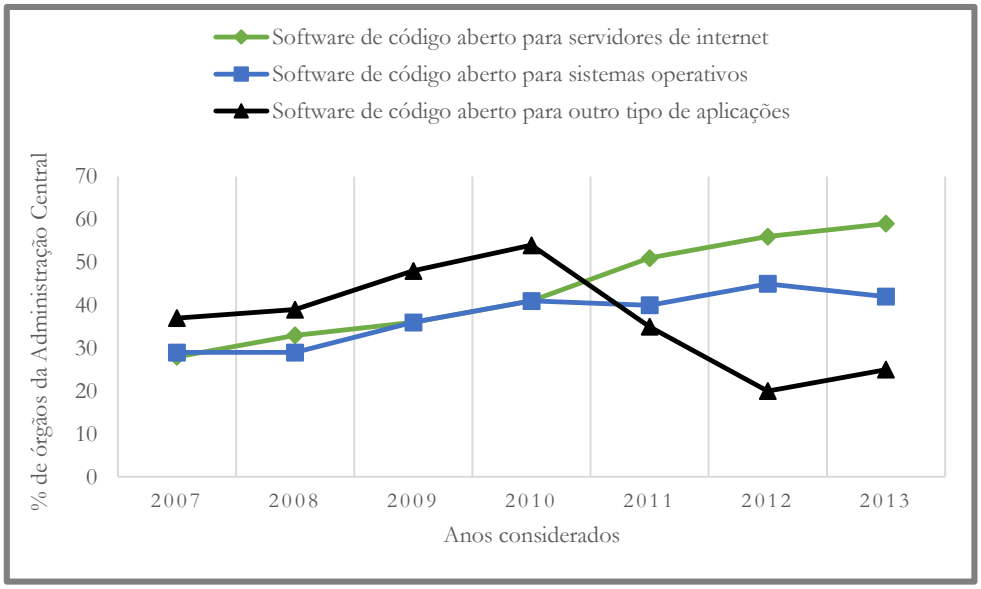


Figura 20 - Percentagem de Organismos da Administração Pública Central por tipo de *software* de código aberto utilizado. Fonte: IUTIC 2013.

O padrão de distribuição temporal de que se falou altera-se substancialmente quando a análise se debruça sobre o SL/CA para outro tipo de aplicações, nas quais se incluem certamente os de SIG/TIG. Nesta linha de evolução podemos individualizar três momentos: i) entre 2007 e 2010, em que se registou um aumento; ii) desde 2010, marco que evidencia que este indicador entrou em declínio – entre 2010 e 212, verificou-se uma redução do uso de SL/CA em 34%; iii) voltando a aumentar em 2013 (5%). As razões para este comportamento errático deste parâmetro de análise não são nem claras, nem objectivas, porque não são transparentes, ainda assim a dedução imediata é a de que a alternativa implementada não tenha obtido resultados satisfatórios.

A Figura 21 denuncia uma situação extremamente similar quando nos debruçamos sobre os dados relativos ao poder local, ou seja, assistimos a uma tendência para o aumento da utilização de SL/CA para sistemas operativos e para servidores de internet, salvo excepções pontuais (2008 e 2009 no caso do primeiro), e verificamos a existência de um decréscimo da utilização de SL/CA para outras aplicações desde 2010, com uma quebra de 30% entre 2010 e 2012. Neste caso, no que diz respeito aos sistemas operativos e servidores, as percentagens são ainda mais expressivas que as apresentadas no gráfico anterior, revelando que uma significativa parte da administração local (mais de 75%) usa esta tecnologia, o que, afinal, até vem provar a sua robustez, qualidade e benefícios.

Ambos os gráficos revelam, no entanto, que as tecnologias livres para sistemas operativos e servidores são as mais implementadas, o que poderá querer dizer que outras áreas de aplicação, como os SIG, podem carecer de *software* capaz de resolver problemas específicos destes organismos.

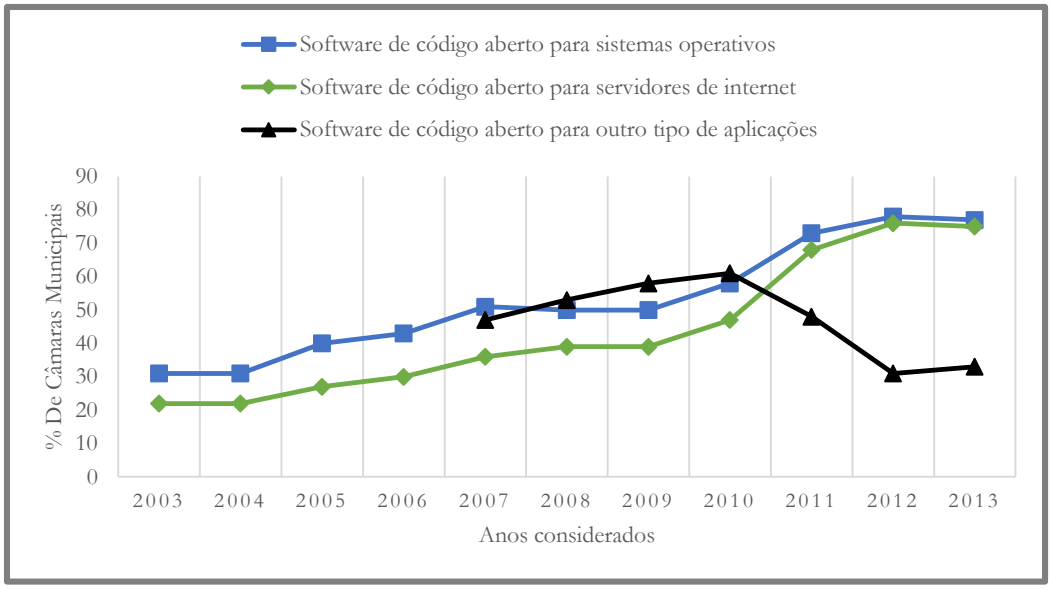


Figura 21 - Percentagem de Câmaras Municipais por tipo de *software* de código aberto utilizado. Fonte: IUTIC 2013.

De qualquer dos modos, estes indicadores revelam que a maior parte dos organismos da APP já se encontra familiarizada com o SL/CA. Contudo, resta saber se a utilização é estrutural ao ponto de sustentar funções críticas ou se obriga a uma complementaridade entre SP e SL/CA.

3.2. Inferência de casos de implementação/migração a partir de informações sobre a aquisição/implementação de *software*

Como observámos antes, os dados do IUTIC não esclarecem quanto a directrizes sobre a implementação de SL/CA para aplicação em áreas como as TIG/SIG. Por isso, e tal como tentámos demonstrar na explicação dos aspectos matriciais da metodologia utilizada neste trabalho, a execução ordenada dos Programas 6, 7, 8 e 9 concretiza uma análise que, a partir BD usada para quantificação das despesas relacionadas com SP e SL/CA, procura localizar casos de implementação à escala concelhia, e, em última instância, especificar alguns órgãos que possam ser tomados como exemplos de boas práticas, contrariando a falta de informação por nós já relatada.

A Figura 22 gerada pelo Programa 6[98], identifica os concelhos do território continental português por tipo de despesa. A Figura 22-A tem como finalidade representar a distribuição de despesas

[98] **Ver Anexo A, p. A4.**

com SP (SIG e S4OS) – apesar de referir também despesas com SL/CA (S4OS e SIG)[99] -, e reúne um conjunto de informações interessantes sobre o período entre 2008 e 2015, que nos permitem destacar as seguintes constatações:

i. Em 21,94% dos concelhos existem organismos públicos que, no que diz respeito ao SP, adquiriram apenas S4OS, ou seja, ou não utilizam quaisquer *software* SIG ou recorrem a soluções livres (tendo despesas, ou não, com esta solução). Por exemplo, tanto Torre de Moncorvo como o Mogadouro ou Idanha-a-Nova registam também despesas com SL/CA SIG. Por outro lado, Alcácer do Sal e Montalegre são exemplos de concelhos onde se adquiriu S4OS proprietário, não havendo quaisquer custos com *software* SIG, nem livre, nem proprietário.

ii. Em 15,83% dos concelhos registaram-se despesas somente com SIG, portanto, tratam-se de organismos que utilizam tecnologias livres para outras aplicações (e. g. Vila Franca de Xira, Coruche, Pinhel, entre outros). Estas instituições, à partida familiarizados com tecnologia de código aberto, provavelmente acharam que as alternativas SIG disponíveis não respondem às suas pretensões.

iii. Em 34,17% dos concelhos existem despesas tanto com S4OS e SIG.

iv. Dos restantes 28,06% dos concelhos, em 23,02% deles, os seus organismos públicos não tiveram qualquer gasto com aquisição de *software* (e. g. Gouveia, Belmonte, Celorico da Beira, entre outros); os restantes 5,04% são concelhos com despesas apenas com serviços relacionados com SL/CA (e. g. Arganil, Manteigas, Espinho, Oliveira do Hospital, entre outros).

Ao contrário da Figura 22-A, a Figura 22-B debruça-se em exclusivo sobre o *software* SIG (SP e SL/CA), mostrando-nos que em 50% dos concelhos, os organismos públicos, ou usam SL/CA (em exclusivo), ou então encomendam os projectos relacionados com informação geográfica a entidades externas. Desta metade, 7,19% tiveram despesas com a aquisição de serviços relacionados com tecnologia livre. Esta figura revela também que em 11,51% dos concelhos se verifica a existência de despesas com SP SIG e SL/CA SIG. Nos restantes (33,49%) utiliza-se em exclusivo SP SIG.

[99] Este cartograma tem como finalidade representar a repartição de despesas com SP (SIG e S4OS), não considerando, para já, gastos no mesmo concelho com SP e SL/CA. Assim, as despesas com SL/CA (S4OS e SIG) incluem-se neste cartograma para diferenciar estes concelhos daqueles que não apresentam quaisquer gastos.

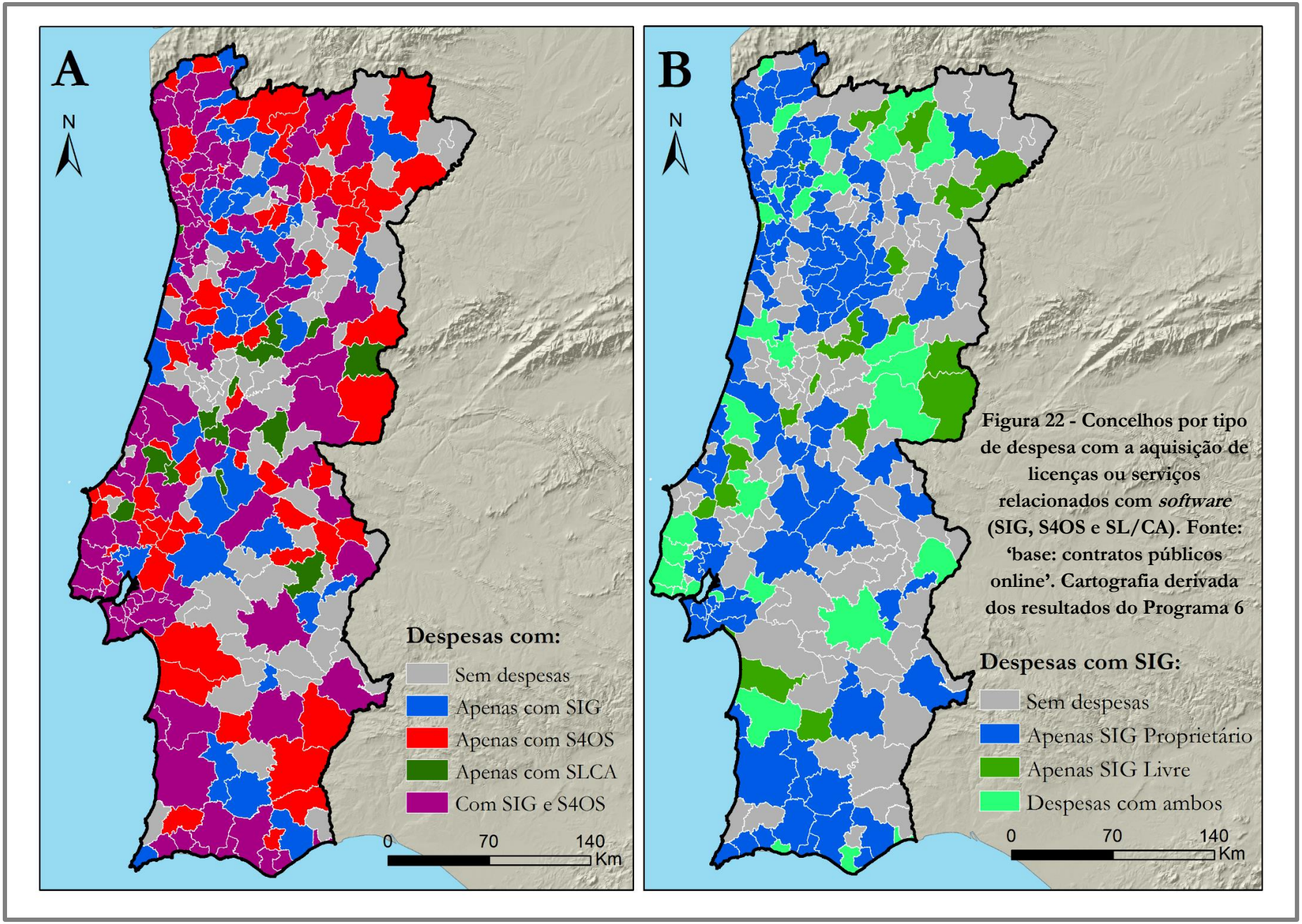


Figura 22 - Concelhos por tipo de despesa com a aquisição de licenças ou serviços relacionados com *software* (SIG, S4OS e SL/CA). Fonte: 'base: contratos públicos online'. Cartografia derivada dos resultados do Programa 6

Como se pode observar, os cartogramas da Figura 22 destacam no espaço geográfico potenciais casos de implementação de SL/CA, em concelhos onde não houve quaisquer despesas com aquisição de SP, onde se estabeleceram contratos com prestadores de serviços relacionados com a implementação/suporte de/a tecnologia livre, ou onde se estabeleceram contratos com fornecedores de SP e com aqueles que prestam suporte a SL/CA.

No entanto, em algumas destas unidades pode não haver organismos que contrataram serviços relacionados com os dois tipos de *software* - embora tenham sido registados contratos para o SP e SL/CA, estes podem ter sido estabelecidos por instituições diferentes.

Face ao exposto, complementamos a ilustração anterior com a cartografia da Figura 23, que é o resultado da execução do Programa 7[100]. Este último implementa um algoritmo que contabiliza, para cada um dos concelhos onde se registaram os dois tipos de contratos, o número de organismos que estabeleceram ao longo do período temporal em análise acordos para a aquisição/implementação dos dois tipos de *software*.

A Figura 23-A discrimina os 31 concelhos que respeitam a condição estabelecida pelo Programa 7, entre os quais se destaca Lisboa (45 organizações que podem ter migrado ou que estão a usar SP e SL/CA em simultâneo), seguida do Porto (com 9). Por seu turno, a Figura 23-B mostra que existem 22 concelhos no território continental onde se localizam as sedes de organizações que adquiriram/implementaram os dois tipos de *software* SIG, contabilizando-se um total de 23 organismos que eventualmente migraram ou usam as duas soluções em simultâneo.

Como o enfoque da nossa análise está na aquisição/implementação de *software* SIG, debruçamo-nos, daqui em diante, apenas nestas 23 organizações. Estas foram identificadas pelo Programa 8[101], a partir do qual foi criada uma nova tabela que reúne as informações contratuais de todos os contratos estabelecidos por estas entidades públicas, nas quais se reúnem também outras duas oriundas dos arquipélagos da Madeira e Açores. A nova tabela é o objecto de entrada do Programa 9[102], que nos indica, destes organismos, quais terão sido os que provavelmente migraram para *software* livre, tendo por base a comparação do número de contratos estabelecidos com fornecedores de SP e prestadores de suporte a SL/CA ao longo dos anos em análise, identificando padrões de utilização.

[100] **Ver Anexo A, p. A5.**
[101] **Ver Anexo A, p. A6.**
[102] **Ver Anexo A, p. A7.**

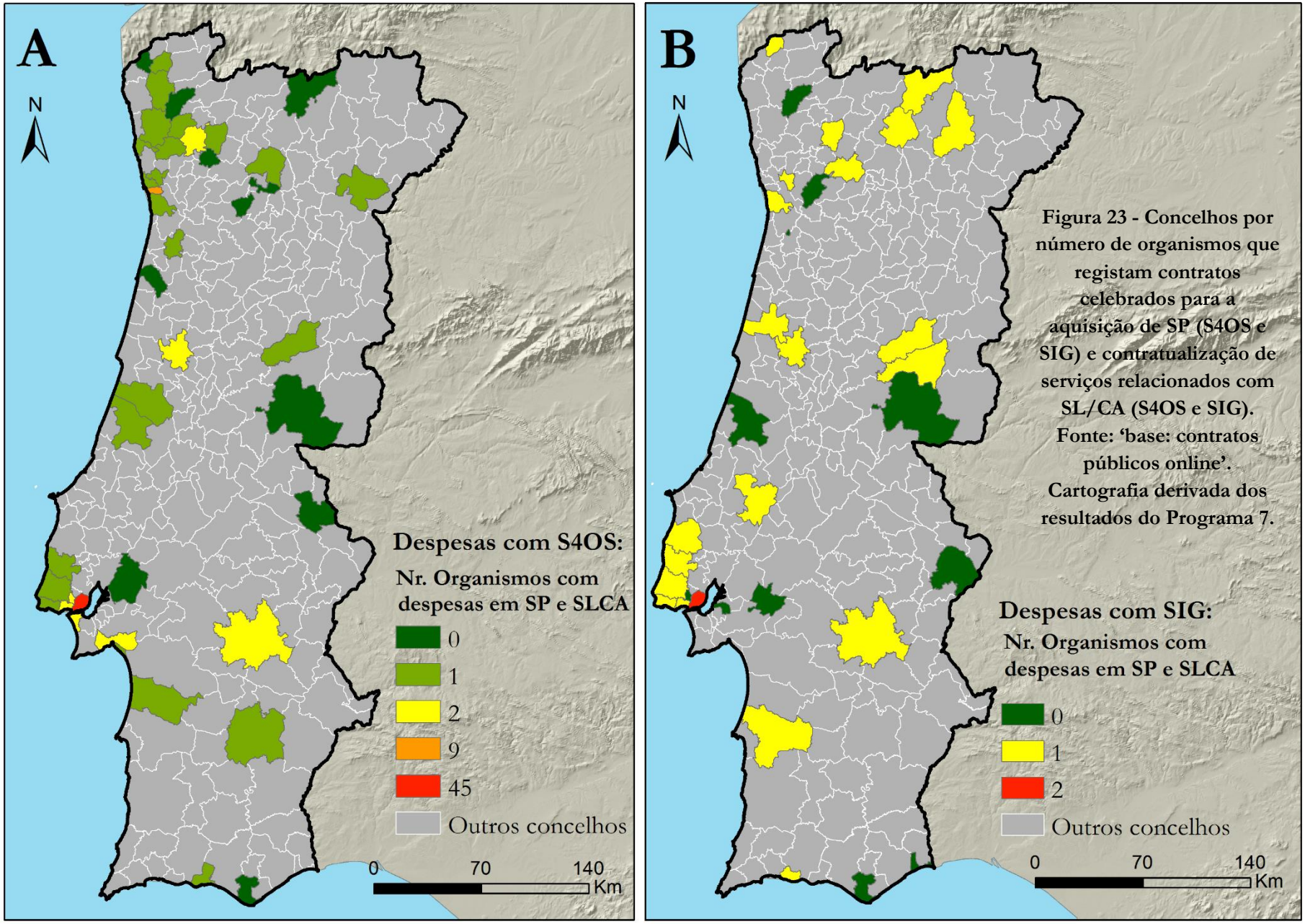


Figura 23 - Concelhos por número de organismos que registam contratos celebrados para a aquisição de SP (S4OS e SIG) e contratualização de serviços relacionados com SL/CA (S4OS e SIG). Fonte: 'base: contratos públicos online'. Cartografia derivada dos resultados do Programa 7.

Não ignoramos as limitações existentes na estrutura lógica deste procedimento (que a análise do seu código revelará), portanto, havendo apenas 25 organismos, analisamos cuidadosamente os resultados, identificando seis instituições que provavelmente migraram e às quais se deverão poder juntar muitas outras que não nos foi possível identificar com esta análise, uma vez que não estabeleceram qualquer contrato, de nenhum tipo.

Optámos por não divulgar o nome dos organismos que resultaram desta análise, pois não nos foi possível confirmar, junto deles, a veracidade das conclusões deste ensaio.

3.3. Recolha de dados sobre adopção de *Software* Livre e de Código Aberto na APP através da disponibilização de um inquérito por questionário

O inquérito anunciado anteriormente tinha dois propósitos centrais:

i. Analisar o estado da adopção de SL/CA SIG na APP;

ii. E recolher relatos e experiências empíricas sobre a migração para SL/CA SIG na APP.

Esta metodologia veio a revelar-se pouco ajustada ou útil para alcançarmos os dois objectivos propostos: por um lado, validámos apenas 45 respostas num universo de 140 inquéritos que nos foram submetidos, número que faz da nossa amostra pouco significativa em termos estatísticos, uma vez que a população que nos interessa (funcionários da APP que trabalham directamente com os SIG) excede os milhares; por outro lado, porque as respostas, que solicitavam ao inquirido um comentário mais pormenorizado sobre a situação do organismo que representa (e. g. experiência vivida em termos de adopção), ou foram demasiado ambíguas (não permitindo extrair dados substanciais) ou converteram-se em campos em branco – não preenchidos. Ainda assim, não deixaremos de apresentar os dados mais relevantes que se podem extrair da análise das respostas ao questionário.

Começamos por caracterizar os perfis dos inquiridos, nos quais se percepciona uma heterogeneidade de habilitações académicas, abrangendo áreas do saber como a geografia (riscos, protecção civil, geomorfologia, ordenamento do território), geologia, urbanismo, engenharia geográfica, ciências e engenharia ambiental, e especializações em detecção remota e SIG. Apesar desta diversidade, a grande maioria (95%) diz ter experiência superior a três anos na área das TIG, e 82% afirma já ter tido experiências de utilização de SL/CA SIG.

No geral, trata-se de indivíduos que representam organismos públicos e que têm conhecimento sobre a existência de soluções alternativas ao SP, no entanto, 9 indivíduos (20%) assumem que o

seu instituto ainda não implementou qualquer solução SIG livre ou de código aberto, apesar da sensibilização para a migração ser transversal a 78% destes organismos. A indisponibilidade de recursos humanos pode ser um condicionalismo à implementação, pois uma percentagem significativa (44%) dos inquiridos afirma que a sua organização não dispõe de pessoal qualificado, contra 53% que disseram o contrário; os restantes - 3%, não responderam. A falta de interesse do pessoal ao serviço (demonstrada por 56% dos inquiridos), traduzida pela cristalização de procedimentos e inércia, é outro dos condicionalismos que não se pode deixar de referir.

Pelo contrário, 80% dos sujeitos declaram que o seu organismo já implementou alternativas ao SP, e destes, apenas 20% se dizem totalmente satisfeitos com a solução encontrada, mas a maioria (51%) admite não estar totalmente satisfeita embora o balanço global seja positivo. Para além dos 20% que não responderam, 7% dos inquiridos expressa uma satisfação parcial (com balanço negativo), e 1 indivíduo (2%) diz-se totalmente insatisfeito.

Claro está que, destas experiências resulta um balanço que apesar de tudo acaba por ser positivo, apesar de se dever sublinhar a questão que se prende com os aspectos que causam descontentamento, informações que não são claramente explicitadas, suscitando dúvidas óbvias – ficamos sem saber se a implementação não teve os resultados esperados devido à ineficiência da tecnologia perante os objectivos da organização, ou se é consequência da incapacidade de a usar correctamente.

Como complemento final da leitura dos dados, não se pode deixar de referir que em 31% dos casos o SL/CA é utilizado para resolver problemas considerados "menores" (menos relevantes e críticos); pelo contrário, 38% referem que usam esta tecnologia na resolução de problemas centrais no funcionamento da organização, e 31% não responderam. Este é um factor importante, pois a aplicação desta tecnologia a funções secundárias implica necessariamente dar continuidade à aquisição de SP.

No final, contabilizam-se 16 (38%) indivíduos que arriscam afirmar que o seu organismo é um caso de sucesso, em oposição aos 10 (22%) que advogam o insucesso do seu processo de migração. Os restantes 42% não responderam, calculamos nós que tenham tido dificuldade em estabelecer uma concepção dos critérios que um exemplo de migração tem de respeitar para ser tratado como caso de sucesso.

Em suma, daqui não podemos retirar outra conclusão que não seja a de que cada caso é um caso, e que estes dados, embora mereçam reflexão, não podem ser generalizados, uma vez que a amostra tem um significado estatístico limitado gerando assim alguma incerteza difícil de esclarecer. Ainda

assim, embora não seja possível contrapor o sucesso com os objectivos da organização, não se devem ignorar os casos assumidos como sendo "de sucesso", demonstrando a capacidade que este tipo de *software* tem potencial para responder às metas estabelecidas, mesmo de tarefas mais complexas.

De igual modo, não se devem também ignorar as referências dos inquiridos a problemas estruturais, o que, na nossa opinião, dá força à ideia de que os decisores, no normal cumprimento das suas funções e responsabilidades, são obrigados a estabelecer formalmente uma estratégia de migração e a comparar o custo total das soluções. Esta assumpção tem fundamento na constatação de situações em que:

i. Se comprova a fraca robustez técnica das soluções de código aberto;

ii. A ausência de assistência técnica adequada e profissional é também uma realidade;

iii. A falta de conhecimento/formação sobre SL/CA e a falta de sensibilização dos utilizadores existem e são factuais;

iv. A necessidade de conhecimentos avançados (por exemplo, de programação), para alguns *software* específicos, também pode condicionar a adopção, intensificando o medo da mudança e a relutância em mudar por ser algo que consume tempo e obriga a um esforço acrescido;

v. A inexistência de meios técnicos dificulta a operacionalização do processo de migração.

Se esta for a realidade, para o decisor verificar se a sua organização se insere no contexto desta realidade, é indispensável que se avalie o custo total do *software* e se adoptem estratégias que contornem as adversidades, nomeadamente em questões relacionadas com o pessoal ao serviço visto que, tal como demonstram os relatos recolhidos, a cristalização de procedimentos, o comodismo, a baixa sensibilidade para a mudança é real e deve ser combatida. Só assim se evitam os chamados casos de insucesso.

Em jeito de síntese, depois desta análise, que balanço se pode fazer sobre o estado de adopção de SL/CA na APP? Ainda que genéricos em demasia e escassos em quantidade e conteúdo, os dados obtidos por esta via permitem concluir que a utilização de SL/CA para sistemas operativos e gestão de servidores já está enraizada e generalizada na APP. Quanto aos SIG, fora os casos de boas práticas que já conhecíamos antes da realização deste trabalho, estamos ainda incontornavelmente reféns de alguma especulação. Aparentemente, o SL/CA SIG parece-nos ser já bastante utilizado na APP, sobretudo para armazenamento, acesso, inquirição e disponibilização de informação

geográfica para dentro e fora da organização, tendo sido possível, em 3.2 e 3.3, listar um conjunto de instituições públicas que recorrem já ao SL/CA SIG no seu dia-a-dia.

Estes exemplos (uns mais incertos do que outros) são prova de que é possível produzir bons resultados com *software* SIG que não seja proprietário, comprovando assim a robustez e fiabilidade do SL/CA para resolução de problemas espaciais específicos. Se é mais dispendioso ou não, isso já é outra questão.

Assim, antecipando desde já trabalho para o futuro, resta-nos direccionar esforços para contactar pessoalmente as organizações identificadas como possíveis casos de sucesso, indagando, junto dos próprios intervenientes no processo de adopção, sobre a veracidade do seu sucesso, no sentido de coleccionar um conjunto de verdadeiras experiências empíricas, indispensáveis à resolução dos problemas estruturais que obstruem o caminho em direcção à implementação, com sucesso, de soluções mais baratas, rentáveis e confiáveis. Este é o passo chave para a constituição da SIGósAPp, cujo conceito apresentaremos em fase mais adiantada deste estudo.

4. Jangada de SIG 2.0 – quantificação de custos e a urgência de uma abordagem multifacetada para disponibilização de um guião/manual (plataforma) para suporte à decisão de migração na Administração Pública Portuguesa

> *"O Software de Código Aberto (SCA) sofreu uma metamorfose para (…) uma forma viável de comercialização. A popular metáfora do bazar, com a qual se tem associado o seu desenvolvimento, mudou para uma metáfora melhor enquadrada num modelo de entrega e suporte do produto, o SCA 2.0."*
>
> Brian Fitzgerald (2006:587)[103]

Tal como antes se referiu, a letra dos últimos OE (e de outros diplomas) impõe a utilização de SL/CA quando esta não põe em causa o normal funcionamento do organismo, tendo em vista a redução das despesas e optimização da interoperabilidade (adopção de normas abertas). Não obstante, os documentos aludidos referem que poderá ser utilizado SP quando fôr fundamentadamente demonstrada a inexistência de soluções alternativas em SL/CA ou quando o custo total da migração fôr superior à aquisição/renovação de licença proprietária.

Uma qualquer equação de determinação do custo total de um determinado SL/CA baseada apenas no custo com o licenciamento (em euros) seria superficial e, por isso incorrecta, na medida em que existem diversos domínios envolvidos - de difícil valoração. Ainda assim, a quantificação das despesas contratualizadas pela APP com a aquisição de licenças de SP ou com a prestação de serviços/suporte relacionados com SL/CA, e a sua inevitável comparação parecem-nos ser o ponto de partida para uma abordagem multifacetada que valorize todos os parâmetros envolvidos na fórmula de cálculo do TCO e do ROI.

Assim, para além das leituras e interpretações que pretendemos sugerir sobre os gastos excessivos da APP com *software* e sobre qual o modelo de *software* mais dispendioso, estes dados e a sua análise

[103] No original, *"I contend that the open source software phenomenon has metamorphosed into a more mainstream and commercially viable form (…). I illustrate this transformation using a framework of process and product factors, and discuss how the bazaar metaphor, wich up to now has been associated with the open source development process, has actually shifted to become a metaphor better suited to the OSS 2.0 product delivery and support process". In* FITZGERALD, Brian (2006) – "The Transformation of Open Source *Software*". **MIS Quarterly**, Vol. 30, N.º 3, 587-598.

fornecem-nos, como se verá mais adiante, informação indispensável ao funcionamento da plataforma SIGósAPp.

Ignorando, por hora, o papel destes dados na orgânica da aplicação, debruçamo-nos sobre o seu significado. A informação da plataforma 'base: contratos públicos online' (adiante designada apenas por 'base') suscita variadas questões; por exemplo:

i. Apesar das imposições legislativas, porque é que ainda se gasta tanto com SP?

ii. Justificar-se-á essa realidade porque a migração é um processo longo e gradual no tempo, pelo que ainda se utilizam soluções proprietárias em paralelo?

iii. Serão os custos com SL/CA de tal modo elevados que inviabilizam a sua adopção?

iv. Será que esta solução não tem ainda robustez e credibilidade suficiente para satisfazer as necessidades de muitos dos utilizadores da APP?

v. Por outro lado, em que medida o desinteresse do pessoal ao serviço nas organizações públicas contribui para a não adopção de SL/CA?

Independentemente das respostas, qualquer uma destas questões levanta o problema advindo da incapacidade de as instituições governamentais fazerem aplicar as leis. Se não for isso, dadas as normas em vigor, parece-nos ser possível a manipulação do processo, conforme os interesses de quem submete o parecer à AMA, ou seja, mesmo que os pareceres e os balanços TCO sejam feitos, pois existe a possibilidade de aquilo que neles consta ser uma deturpação da realidade, uma análise parcial em que se tenta a todo o custo arranjar justificações para se usar o *software* que se pretende, embora possa existir outro mais vantajoso em múltiplos sentidos.

As próprias forças políticas parecem dar-nos garantias de que existe fiscalização destes processos, no entanto, os avultados montantes registados, sobretudo no que concerne aos SIG e TIG, são completamente desprovidas de razão ou justificação, uma que vez que está mais do que comprovada a qualidade do SL/CA na resolução, se não de todos, pelo menos de uma ampla gama de problemas reais.

Podemos dissertar e problematizar o que quisermos sobre estas questões e dúvidas que encerram, contudo, no momento, não passariam de conjecturas e opiniões, na medida em que só a averiguação, no local, do que realmente se passa, nos daria respostas assertivas e minimamente fiáveis. Por agora, debruçamo-nos sobre os dados que denunciam quanto se gastou com *software*, uma realidade que deve merecer, por parte de cada um de nós, uma profunda reflexão.

Assim, no que diz respeito ao SP (SIG e S4OS), considerámos um total de 13 Empresas (7 do sector dos SIG e 5 fornecedores de *software* de produtividade/escritório e outros – S4OS) tendo por base a análise de 3180 contratos (887 relativos aos SIG e os restantes ao S4OS)[104] estabelecidos entre 2008 e 2015. Tendo em vista a obtenção de um termo de comparação, contabilizaram-se, para o mesmo intervalo de tempo, as despesas com a contratualização de serviços relacionados com SL/CA, num total de 19 empresas (2 de SIG, 17 S4OS) e 419 contratos (dos quais 99 se referem a *software* SIG) [105].

Numa primeira leitura dos valores em jogo podemos adiantar que, entre 2008 e 2015, o Estado Português gastou 37 981 809,91 €, apenas em SP SIG. Se incluirmos os valores relativos ao S4OS, o montante ultrapassa os 250 000 000 € (Tabela XIII).

A observação atenta dos valores envolvidos não deixa margem para dúvida, quanto ao peso que, os valores totais envolvidos nos contratos de aquisição e/ou manutenção de serviços informáticos para as diversas áreas na APP, representa para o Erário Público.

Numa primeira reflexão sobre os dados (Tabela XIII), não podemos deixar de assinalar as discrepâncias entre 2008 e 2009, que acentuam uma assinalável subida (em flecha) das despesas – aproximadamente 850% para o *software* SIG e 1000% no caso do S4OS.

Estamos em crer, porém, que esta disparidade não é absolutamente linear e lida apenas desta forma, visto que a plataforma 'base' iniciou os seus trabalhos a 30 de Julho de 2008 (http://www.base.gov.pt/Base/pt/CodigoDosContratosPublicos/Antecedentes, acedido em 20 Maio, 2015), evidência de que a estão em falta inúmeras instâncias nesse ano, justificando, assim, as diferenças sinalizadas.

4.1 Despesas com *Software* SIG Proprietário

Os SIG desempenham um papel crucial na sociedade sem que as populações se apercebam, ainda em pleno das suas potencialidades talvez porque, curiosamente, também não tomaram ainda consciência dos avultados gastos do Estado com este tipo de *software*.

[104] Tenhamos a certeza de que os contratos estabelecidos com as empresas seleccionadas não representam, por ventura, a maior parte dos gastos que a APP e o Estado Português têm com *software*. Infelizmente, no presente trabalho conseguimos apenas pôr em evidência os montantes relativos a estas instituições, esperando que no futuro, nos seja possível alargar a pesquisa.

[105] Como já se disse, os dados apresentados resultam de um levantamento minucioso efectuado sobre uma base de dados pública oficial ('base'), optou-se pela omissão dos nomes apenas por uma questão de ética, até porque a sua menção não relevaria para a análise dos dados nem para a explicação dos resultados obtidos.

Tabela XIII - Despesas de entidades públicas com *Software* Proprietário (SIG e S4OS) em Portugal, por ano e por empresa produtora. Fonte: 'Base: contratos públicos online'

	Organismo Instituição	Despesas (em €)								
		2008	2009	2010	2011	2012	2013	2014	2015	Total
Software SIG	Empresa A	0,00	57 764,75	17 670,00	47 125,50	79 567,70	34 714,76	23 627,93	18 756,30	**279 226,94**
	Empresa B	0,00	14 000,00	0,00	20 290,00	0,00	0,00	0,00	0,00	**34 290,00**
	Empresa C	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
	Empresa D	269 164,58	2 900 487,44	4 079 129,54	2 733 780,36	3 100 343,19	1 272 361,58	2 018 622,25	1 522 275,63	**17 896 164,57**
	Empresa E	176 776,92	1 146 249,78	1 430 581,56	2 410 357,47	684 966,78	562 093,81	1 241 159,99	7 381 867,13	**15 034 053,44**
	Empresa F	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,00**
	Empresa G	34 480,72	413 927,41	634 624,68	913 966,78	648 628,30	643 171,99	719 874,52	729 400,56	**4 738 074,96**
	Total (SIG)	**480 422,22**	**4 532 429,38**	**6 162 005,78**	**6 125 520,11**	**4 513 505,97**	**2 512 342,14**	**4 003 284,69**	**9 652 299,62**	**37 981 809,91**
S4OS	Empresa V	899 974,68	26 471 745,03	15 988 105,25	12 206 495,53	20 313 896,27	12 278 163,75	25 333 715,04	30 387 570,05	**143 879 665,00**
	Empresa W	72 845,91	248 268,48	243 575,49	229 977,03	190 565,97	186 446,68	440 108,27	122 433,37	**1 734 221,20**
	Empresa X	32 306,40	628 251,50	507 956,80	605 865,00	378 623,47	375 216,93	492 047,62	116 463,40	**3 136 731,12**
	Empresa Y	0,00	20 000,00	56 930,00	15 137,00	47 135,00	70 103,00	56 919,90	226,00	**266 450,90**
	Empresa Z	1 854 591,78	6 489 339,45	6 046 905,71	7 690 584,51	4 701 393,57	15 269 160,03	11 925 490,53	10 070 666,45	**64 048 132,03**
	Total (S4OS)	**2 859 718,77**	**33 857 604,46**	**22 843 473,25**	**20 748 059,07**	**25 631 614,28**	**28 179 090,39**	**38 248 281,36**	**40 697 359,27**	**213 065 200,85**
	SIG +S4OS	**3 340 140,99**	**38 390 033,84**	**29 005 479,03**	**26 873 579,18**	**30 145 120,25**	**30 691 432,53**	**42 251 566,05**	**50 349 658,89**	**251 047 010,76**

Tabela XIV - Frequências que enunciam o número de contratos compreendidos em intervalos de despesa pública com *Software* SIG Proprietário, entre 2008 e 2015. Fonte: 'Base: contratos públicos online'

	Absolutas	Acumuladas	Relativas (%)	Acumuladas (%)
0 – 10 000 (1)	272	272	30,67	30,67
10 000 – 25 000 (2)	258	530	29,09	59,75
25 000 – 50 000 (3)	200	730	22,55	82,30
50 000 – 100 000 (4)	107	837	12,06	94,36
100 000 – 250 000 (5)	37	874	4,17	98,53
250 000 – 500 000 (6)	9	883	1,01	99,55
500 000 – 750 000 (7)	0	883	0,00	99,55
750 000 – 1 000 000 (8)	3	886	0,34	99,89
> 1 000 000 (9)	1	887	0,11	100,00

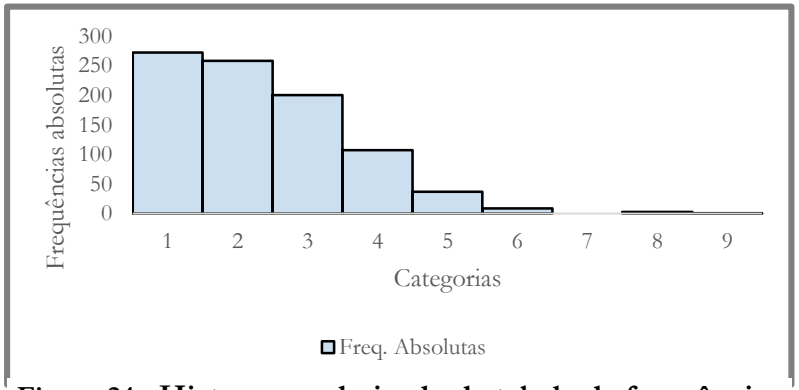


Figura 24 - Histograma derivado da tabela de frequências apresentada na Tabela XIV.

Os (quase) 38 000 000 € resultam do somatório de 887 contratos com valores muito diversos, excedendo quatro vezes os 750 000 € e uma vez o 1 000 000 € (Tabela XIV e Figura 24)[106]. Na maior parte dos casos (82,30%), os valores contratuais não excedem os 50 000 €. De facto, entre os 306 organismos/instituições que têm presença na nossa BD, 59,15% gastaram até este valor (Tabela XV), embora cada um dos 306 tenha despendido, em média, 124 123,56 €. Destaque, ainda, para as 5 instituições que tiveram uma despesa maior que 1 000 000 €.

Tabela XV - Frequências absolutas e relativas que contabilizam o número de organismos públicos por categorias despesa com *Software* SIG Proprietário, entre 2008 e 2015. Fonte: 'Base: contratos públicos online'

	Absolutas	Acumuladas	Relativas (%)	Acumuladas (%)
0 – 10 000	55	55	17,97	17,97
10 000 – 25 000	57	112	18,63	36,60
25 000 – 50 000	69	181	22,55	59,15
50 000 – 100 000	53	234	17,32	76,47
100 000 – 250 000	44	278	14,38	90,85
250 000 – 500 000	17	295	5,56	96,41
500 000 – 750 000	4	299	1,31	97,71
750 000 – 1 000 000	2	301	0,65	98,37
> 1 000 000	5	306	1,63	100,00

De tudo o que se gastou, 57% diz respeito a despesas contraídas entre 2008 e 2012 (inclusive), período de tempo em que não havia controlo da AMA sobre os contratos com a aquisição de *software*. Os restantes 43% (16 167 926,45 €) referem-se ao período subsequente e resultam, ou da incapacidade do SL/CA SIG, ou do custo elevado da migração, ou das condições que fazem com que um contrato não seja alvo do parecer da AMA.

Tal como se disse anteriormente, segundo o Decreto-Lei 107/2012, os contratos abaixo do limiar dos 10 000 € podem ser celebrados sem qualquer controlo por parte da entidade reguladora (AMA). Estes contratos dizem respeito a 30,67% da nossa amostra (272/887), no entanto, só 122 foram assinados entre 2013 e 2015. Face ao exposto, estes 122 contratos e os 773 592,16 € envolvidos (2% do total) não foram alvo de parecer.

A este montante juntam-se os 215 947,10 € que as instituições de ensino superior gastaram, totalizando 989 539,26 € em contratos não controlados pela AMA depois da promulgação do supramencionado diploma.

[106] Todos os quadros de frequência apresentados ao longo desta análise são produto da execução do Programa 1 – Ver Anexo A, p. A1.

A outra fatia do total de despesa entre 2013 e 2015, ou estão em incumprimento, ou não viram no SL/CA uma alternativa viável, ou constituem excepções, podendo beneficiar também, em último caso, da "não resposta" em tempo útil, por parte da AMA ao pedido de parecer. Independentemente das razões que possam justificar estas verbas, parece-nos óbvio que as recentes imposições legislativas não estão a surtir o efeito pretendido. Por um lado, segundo a Tabela XIII, entre 2013 e 2015, os valores em causa representam 43% dos custos totais e, por outro lado, a evolução cronológica dos totais de despesa por ano evidenciam a ineficácia destes documentos legais.

As despesas com *software* SIG, em moldes gerais, evidenciam uma tendência de subida e estabilização de custos que se verificou entre 2009 e 2012, e também a redução substancial do montante em 2013 (ano da entrada em vigor do OE 2013) – em mais de 40% - comparativamente ao ano anterior (Figura 25). Mas, ao contrário do que seria de esperar, em 2013, o que se pensava ser o início de uma tendência para a descida, estranhamente não se confirmou em 2014 e 2015, anos em que os custos subiram em 59,34% e 284% relativamente a 2013, respectivamente.

A monitorização da despesa deve, assim, ser uma das preocupações das entidades com competências nestas questões, motivando que estas tomem medidas em conformidade com esta realidade, que, neste momento, revela a ineficácia dos diplomas cujo objectivo é a redução da despesa do Estado Português.

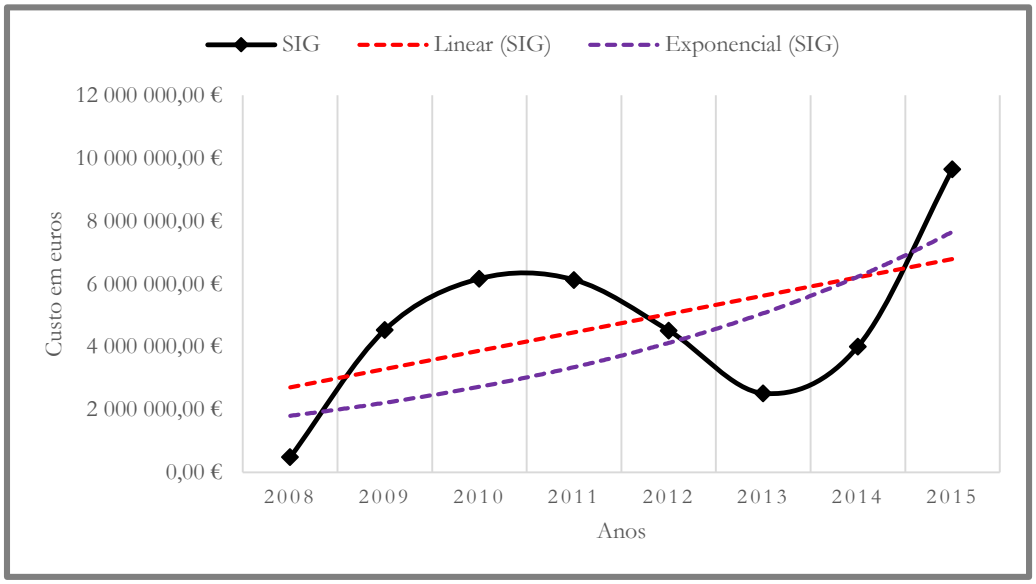


Figura 25 - Variação e tendência dos totais da despesa com SIG Proprietário em Portugal, para o intervalo de tempo em apreciação. Fonte: 'Base: contratos públicos online'.

Numa leitura mais fina, ao nível da liderança por empresa (Figura 26), destaca-se a Empresa D, claramente dominante no mercado português, e que se encontra a recuperar da queda registada em 2013. A Empresa E também merece destaque sobretudo pela hegemonia no ano de 2015, superando a marca dos 7 000 000 €.

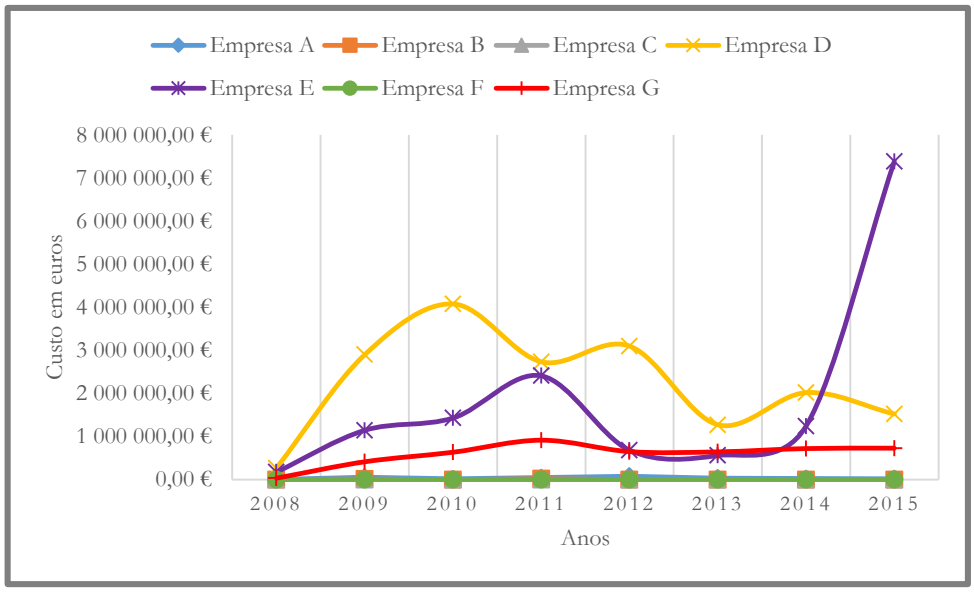


Figura 26 - Quantificação dos gastos de instituições públicas portuguesas com *Software* SIG proprietário, para o intervalo de tempo em apreciação, considerando algumas das principais empresas fornecedoras. Fonte: 'Base: contratos públicos online'.

As restantes empresas revelam um ritmo estacionário, em geral, de reduzida expressão (ou nula expressão, no caso da Empresa C e F), ainda que se possa considerar como excepção a Empresa G, cujos valores envolvidos nos contratos com a APP parecem não ter sentido o abalo introduzido pelo OE de 2013.

Uma simples interpretação da informação disponibilizada na Figura 26, leva-nos a perceber que não se delineia qualquer tipo de correlação, ou de indícios de um comportamento uniforme, nos contratos estatais formalizados com as empresas do sector consideradas. Por exemplo, a Empresa D, embora tenha tido menos contratos em 2010 do que em 2009, apresenta um pico em 2010 (4 079 129 €), a empresa E, por sua vez, tem um pico de vendas em 2011 (2 410 357 €), ano em que se aproxima do valor despendido com *software* da Empresa D, com uma diferença de apenas 323 422 €. Curiosamente, este é também o ano em que mais se gastou com *software* produzido pela Empresa G (913 966 €). Isto aponta para a possibilidade de a distribuição temporal destes valores depender do *timing* escolhido por cada empresa para o lançamento de novas versões dos seus

programas (que se traduz numa corrida à renovação de licenças nos momentos de novos lançamentos), e não da promulgação de medidas que condicionam a aquisição de *software*.

A única excepção acontece em 2015, ano em que se lançou a nova versão do *software* da Empresa D sem que houvesse uma tradução em aumento de vendas. A Empresa E constitui outra fuga à normalidade, pois o aumento exponencial das verbas em 2015 deve-se a uma situação pontual em que se formalizou um acordo, cujo preço contratual é de mais de 6 000 000 €.

Os objectivos de redução de despesa estatal não estão a ser materializados, colocando em evidência as carências ao nível do fomento de estratégias que procurem perceber as razões que justificam a ineficácia da legislação que promulga a obrigatoriedade de implementação de soluções de baixo custo quando estas são mais rentáveis que a alternativa proprietária.

A pesquisa por números e valores efectuada sobre a plataforma digital de contratos públicos permitiu, também, organizar os dados por sectores de actividade. A distribuição dos gastos na APP com *software* SIG (Tabela XVI e Figura 27) revelam que os Serviços Municipalizados foram os que mais gastaram (46%), seguindo-se os sectores estatais da Administração Central e Regional (29%) e dos Organismos de Gestão do Território e Recursos (12%). Panorama que, no geral, se mantém se olharmos apenas para o período entre 2013 e 2015, no qual a Administração Central e Regional se evidencia e supera mesmo o volume de negócios dos Serviços Municipalizados (Figura 28).

No pormenor, estes sectores (Serviços Municipalizados e Administração Central e Regional) confirmam a diminuição dos gastos entre 2012 e 2013, sendo o segundo, a par com o da Educação, aqueles que registaram maiores descidas, de 85% e 72%, respectivamente. No entanto, o gráfico da Figura 29 permite destacar a importância do volume de negócios por parte dos Serviços Municipalizados e, se confrontado com os dados apresentados anteriormente, rapidamente nos ajuda a concluir que 2011 foi o ano em que os serviços municipais decidiram que necessitavam de comprar/renovar licenças deste tipo de programa informático, registando-se um decréscimo até 2013, para depois a tendência se inverter em 2014 e 2015, inversão essa ainda mais saliente no que diz respeito à Administração Central e Regional.

Tabela XVI - Despesas de entidades públicas em *Software* SIG Proprietário, para o intervalo de tempo em apreciação, por sector de actividade da administração pública. Fonte: 'Base: contratos públicos online'

	Despesas (em €)								
	2008	2009	2010	2011	2012	2013	2014	2015	Total
Serviços Municipalizados	202 861,10	2 376 202,43	2 106 537,37	4 079 766,73	2 467 061,71	1 590 585,16	1 968 242,75	2 575 865,86	**17 367 123,11**
Saúde	0,00	555 576,88	324 341,43	195 726,05	309 611,74	195 325,00	11 766,08	198 184,42	**1 790 531,60**
Administração Central e Regional	138 221,71	453 233,88	1 476 884,71	837 566,96	926 346,24	139 094,20	638 606,80	6 354 930,11	**10 964 884,61**
Outros Serviços	13 317,60	287 287,30	517 236,13	365 438,83	115 666,37	193 882,08	577 348,08	95 988,75	**2 166 165,14**
Organismos de Gestão do Território e Recursos	85 814,85	783 472,70	1 527 209,61	457 785,99	369 473,69	252 505,10	723 333,58	276 188,30	**4 475 783,82**
Educação	24 537,96	27 940,36	54 262,10	40 701,12	159 594,12	44 229,00	52 915,10	118 803,00	**522 982,76**
Segurança	15 669,00	48 715,83	155 534,43	148 534,43	165 752,10	96 721,60	31 072,30	32 339,18	**694 338,87**
Total	**480 422,22**	**4 532 429,38**	**6 162 005,78**	**6 125 520,11**	**4 513 505,97**	**2 512 342,14**	**4 003 284,69**	**9 652 299,62**	**37 981 809,91**

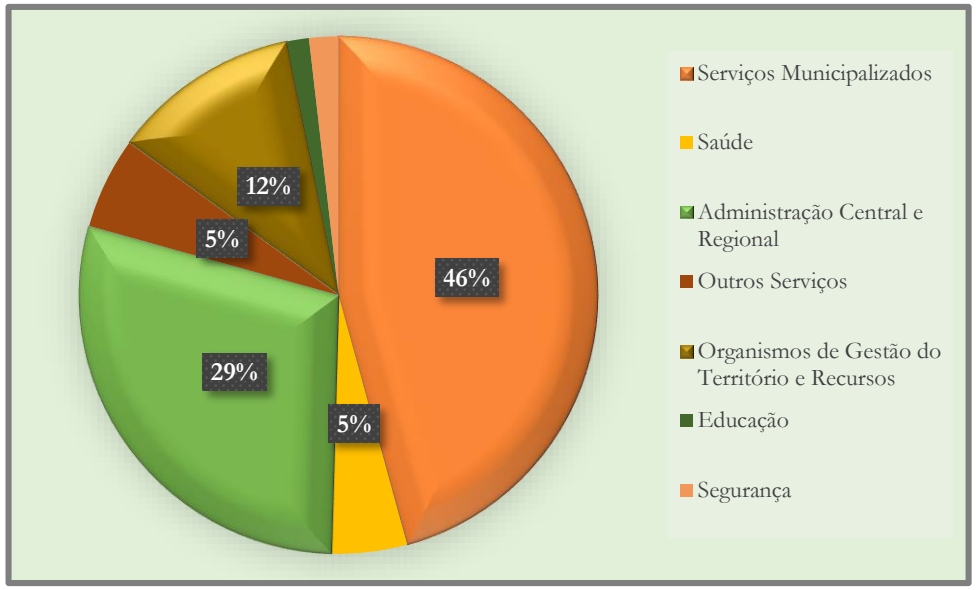


Figura 27 - Distribuição dos encargos de instituições públicas portuguesas com *Software* SIG proprietário, por sector de actividade (2008-2015). Fonte: 'Base: contratos públicos online'.

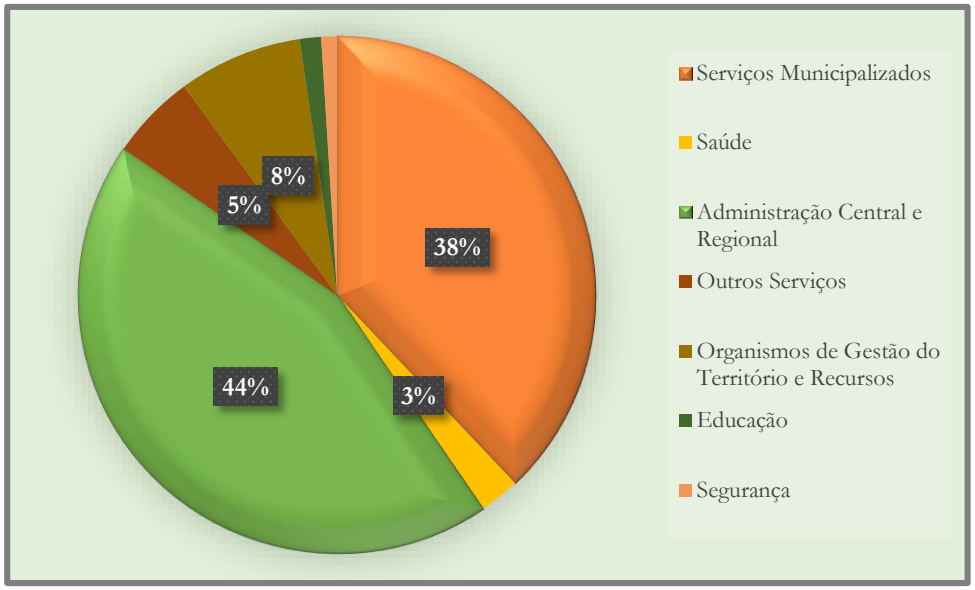


Figura 28 - Distribuição dos encargos de instituições públicas portuguesas com *Software* SIG proprietário, por sector de actividade (2013-2015). Fonte: 'Base: contratos públicos online'.

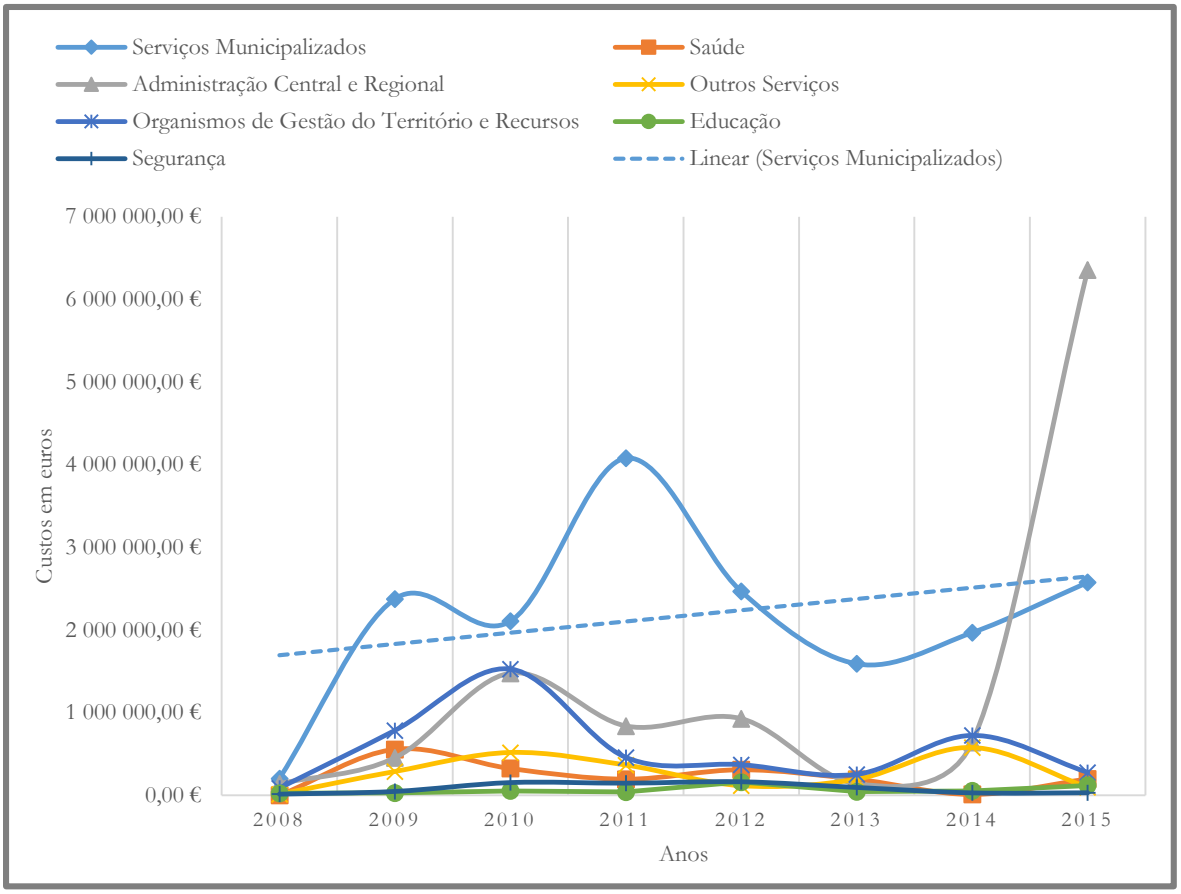


Figura 29 - Distribuição dos encargos de instituições públicas portuguesas com a aquisição de *software* SIG proprietário, por ano e por sector de actividade da APP. Fonte: "Base: contratos públicos online'.

A mudança no sentido de evolução da despesa relativa a estes sectores, nomeadamente a da administração local, deve ser fonte de maior preocupação por parte das entidades com responsabilidades na fiscalização do rigoroso cumprimento das normas, que no exercício do seu dever e competências devem perceber as razões para o recrudescimento de situações observadas no passado.

No que respeita à administração local, julgamos ser da competência dos fiscalizadores a condução destes organismos no sentido da migração, tendo como suporte a experiência de outros municípios que já usam em exclusivo soluções de código aberto e que não surgem na nossa BD. Necessidade que se sublinha ao verificarmos que existem bastantes concelhos do interior do país que, mesmo não se destacando em termos absolutos no total dos custos, compraram este tipo de *software* e gastaram dinheiros públicos em serviços a ele associados, muitas vezes avultados. Tendo em conta as dinâmicas territoriais e as trajectórias de desenvolvimento caracterizadas pela quebra dos quantitativos populacionais (baixa densidade) e regressão do tecido económico (matriz rural), esta

constatação não pode deixar de se envolver em algum polémica e controvérsia, uma vez que as verbas para aí canalizadas poderiam ter tido como opção alternativa um investimento público em rubricas mais urgentes e/ou passíveis de gerar e atrair mais investimento, logo, atrair mais população (Figura 30).

Esta figura[107] apresenta a informação relativa à distribuição espacial das despesas efectuadas por organismos públicos, entre 2008 e 2015, para aquisição de *software* SIG. Para além da evidencia já descrita, realçamos uma polarização dos gastos nas áreas metropolitanas de Lisboa e do Porto, à semelhança de outras variáveis socioeconómicas que demonstram as mesmas assimetrias – só Lisboa e o Porto concentram um volume total na ordem dos 16 391 243 € (43% do total). Não obstante, existe uma avultada diferença entre os dois concelhos mais representativos das referidas conurbações (Lisboa – 15 277 126 €, e Porto – 1 114 116 €), o que se explica pelo facto de se concentrar em Lisboa, a grande maioria das sedes de organismos estatais com competências na coordenação do planeamento e do ordenamento territorial português.

Num outro plano, mas a merecer semelhante destaque, refere-se uma considerável concentração de despesas com SP SIG em áreas mais próximas do litoral entre Lisboa e o norte do país, ainda que não seja uma faixa absolutamente contínua entre Lisboa e Porto. Nesta região, destacam-se as despesas da responsabilidade dos municípios de Coimbra – 994 598 € e Leiria – 614 402 €.

Quanto às regiões insulares, na Região Autónoma da Madeira destaca-se o concelho do Funchal – 582 140 €, e, na Região Autónoma dos Açores, Ponta Delgada – 1 773 443 €, valores que só se conseguem compreender por também aí estarem localizadas as sedes das delegações regionais de organismos da administração estatal.

.

[107] Na construção do cartograma da Figura 30 optámos por uma classificação dos dados com 9 classes, pois, na nossa perspectiva, a atribuição de um menor número de categorias teria como consequência a inclusão na mesma classe, de concelhos com valores astronomicamente díspares. Esta estratégia permite também identificar com maior rigor os concelhos com maior concentração de volume de despesa, bem como demonstrar que a maior parte dos concelhos (70%) não gastou mais que 100 000 € em oito anos e que (86%) apresenta valores abaixo dos 250 000 €. Neste processo de classificação, focamo-nos em garantir que todas as classes tivessem uma representação aceitável, rastreamento feito a partir de uma tabela de frequências gerada pelo Programa 1 **(ver Anexo A, p. A1)** – todas as classes representam entre 10% a 20% dos concelhos.

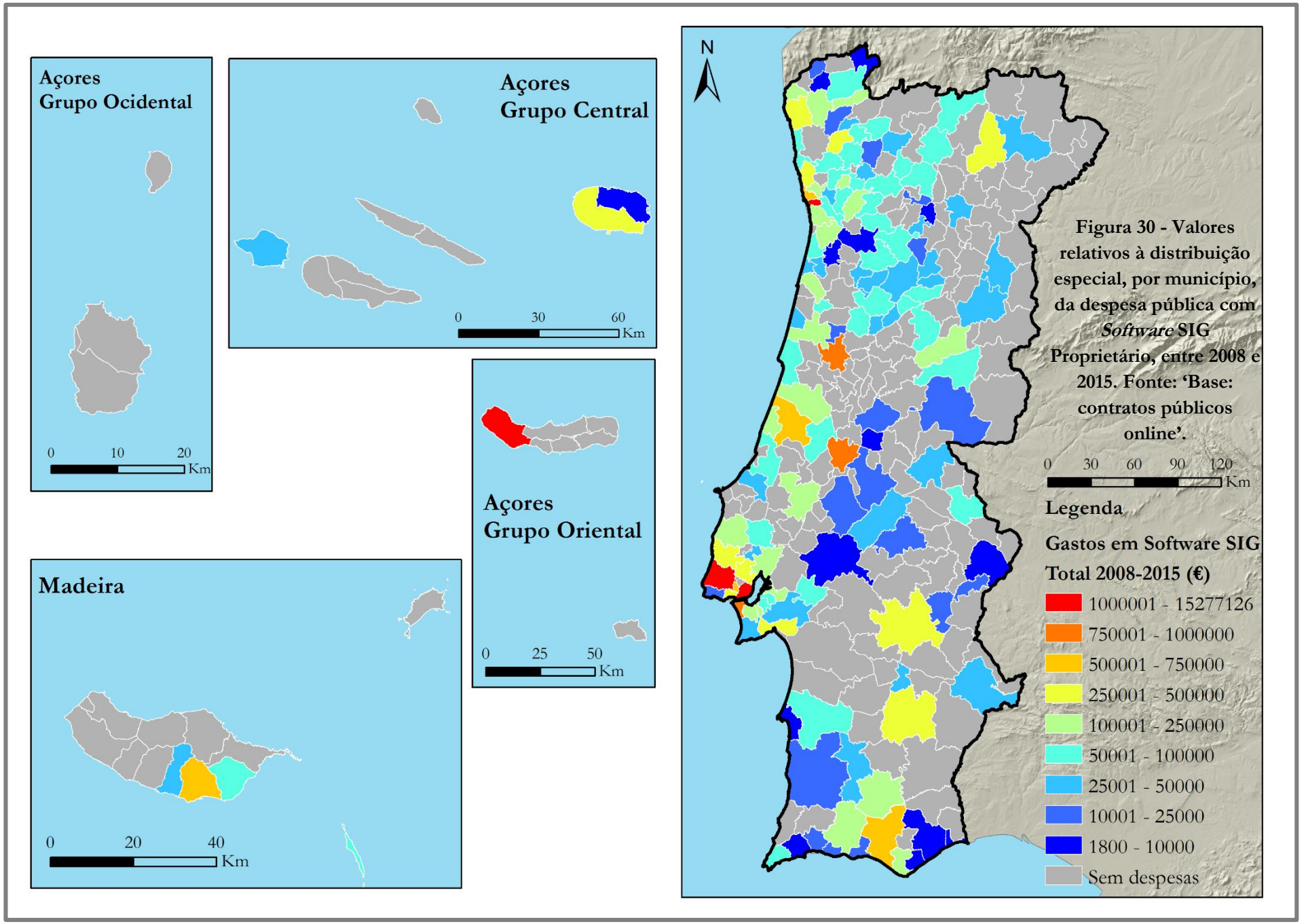


Figura 30 - Valores relativos à distribuição especial, por município, da despesa pública com *Software* SIG Proprietário, entre 2008 e 2015. Fonte: 'Base: contratos públicos online'.

4.2 Despesas com *Software for Operating Systems* – S4OS Proprietário

Os dados disponibilizados na Tabela XIII (cfr. p. 110) documentam valores absolutamente impressionantes. Entre 2008 e 2015, o Estado português despendeu 213 065 200,85 € com *software* de produtividade/escritório, científico, multimédia e de armazenamento e gestão de informação, valor que acaba por ser muito mais significativo do que o apresentado para o *software* SIG. Aquele valor traduz, como se observa na Figura 31, a dependência face a sistemas e programas patenteados pela Empresa V, que ocupa uma posição claramente dominante no mercado, mesmo em contexto mundial, posição que chegou a ser objecto de penalizações pesadas devido a procedimentos ilícitos praticados ao abrigo de um monopólio que esvaziava as possibilidades de implantação de concorrência (Patriarca *et al.*, 2014a).

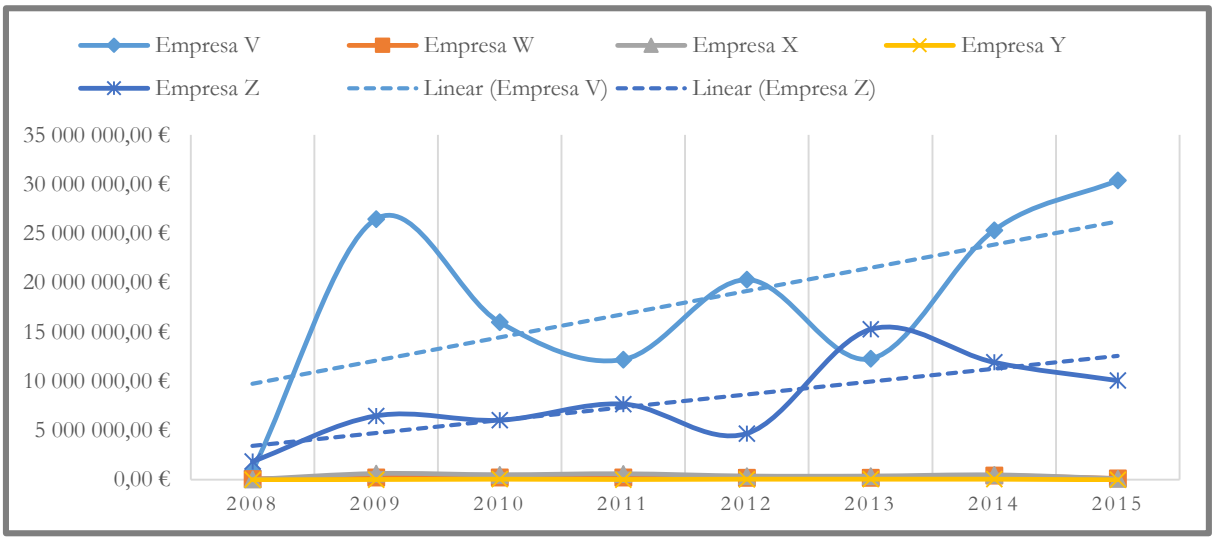


Figura 31 - Quantificação dos gastos de instituições públicas portuguesas com S4OS Proprietário, para o intervalo de tempo em apreciação, considerando apenas algumas das principais empresas fornecedoras. Fonte: 'Base: contratos públicos online'.

A Empresa Z permanece na sombra da anterior, ainda que se destaque das demais com um volume de negócios que ascende aos 64 000 000 €. As restantes, têm menor expressão em Portugal, representando apenas 2,41% das verbas totais.

À semelhança do que se sucede no *software* SIG, desde da promulgação do OE 2013, a tendência para o decréscimo da despesa com S4OS é pontual. Os aparentes efeitos do OE 2013 são visíveis somente para a Empresa V, já que todas as outras registaram aumentos de vendas. Fazendo mais uma vez o paralelo com o *software* SIG, nota-se que a variação dos valores de despesa depende dos

timings de lançamento de novas versões, justificando-se assim os picos registados por esta multinacional em 2009, 2012, 2014 e 2015.

A Empresa Z poderá ser a única a sentir os efeitos do panorama legislativo nacional relativo ao SL/CA, assinalando uma gradual descida desde 2013. A alternativa livre ao *software* desta fornecedora é, desde alguns anos a esta parte, distinguida pelos vários utilizadores como uma solução confiável, robusta e de qualidade (comparável a qualquer SP), ao contrário do que acontece com outros projectos que, pelo seu inferior nível de maturidade, ainda não conquistaram o *main stream* daqueles que utilizam este tipo de solução informática.

Assinalamos novamente o incumprimento dos objectivos a que as políticas que visam a redução da despesa, se propuseram. As verbas gastas pelo Estado Português entre 2013 e 2015 somam 50% do total. Destes 107 000 000 €, pelo menos 8 875 382 € representam gastos que não foram monitorizados pela AMA (7 035 988,55 € relativos a despesas correntes de instituições de ensino superior; 1 553 127,73 € em contratos abaixo dos 10 000 €; 286 265,86 € referem-se a despesas de entidades com estatuto de independência administrativa).

Ao derivarmos esta análise para uma comparação entre os dados por diferentes sectores de actividade da APP, percebe-se que a classe "Outros Serviços" (conjunto de serviços implementados por empresas com alguma autonomia relativamente à máquina central do Estado e que nada têm a ver com a gestão do território) faz-se notar pelos maiores gastos comparativamente a outros sectores (33%), acompanhados pelos 27% da Administração Pública Central e Regional e 15% dos Serviços Municipalizados (Tabela XVII e Figura 32). Os dois primeiros, para além das suas percentagens, revelam, nos últimos dois anos, uma tendência para a subida (Administração Central de Regional -438%, e Outros Serviços – 80,71%), tal como se pode observar na Figura 33. Devem, por isso, ser alvo de supervisão atenta por parte das entidades com responsabilidades nos processos de revisão dos pedidos de parecer que lhe vão sendo enviados, contornando possíveis irregularidades, ou imperfeições no sistema de avaliação do custo da solução informática a adquirir, ou na composição normativa dos diplomas que têm vindo a ser promulgados.

Tabela XVII - Despesas de entidades públicas em S4OS Proprietário, para o intervalo de tempo em apreciação, por sector de actividade da administração pública. Fonte: 'Base: contratos públicos online'

	Despesas (em €)								
	2008	2009	2010	2011	2012	2013	2014	2015	Total
Serviços Municipalizados	38 832,63	3 944 380,58	4 181 732,56	3 583 651,48	5 245 113,04	5 769 536,96	5 241 079,78	3 571 046,98	**31 575 374,01**
Educação	0,00	1 261 419,15	90 690,50	878 087,98	887 253,19	7 426 577,61	2 816 237,81	3 353 382,23	**16 713 648,47**
Administração Central e Regional	774 283,12	18 393 139,36	3 539 933,31	2 339 016,56	4 105 377,49	2 865 798,37	9 606 391,33	15 420 209,86	**57 044 149,40**
Organismos de Gestão do Território e Recursos	767 542,76	5 822 144,61	11 559 505,55	7 898 824,37	8 360 572,50	8 202 118,70	13 616 819,06	14 821 758,17	**71 049 285,72**
Saúde	615 532,76	1 727 332,51	1 382 651,00	4 138 469,76	4 644 454,37	1 138 289,76	4 837 176,98	1 218 288,06	**19 702 195,20**
Segurança	464 452,51	1 564 522,73	1 670 404,55	1 877 875,38	2 221 546,85	2 696 209,19	2 027 105,40	2 312 673,97	**14 834 790,58**
Outros serviços	199 074,99	1 144 665,52	418 555,78	32 133,54	167 296,84	80 559,80	103 471,00	0,00	**2 145 757,47**
Total	**2 859 718,77**	**33 857 604,46**	**22 843 473,25**	**20 748 059,07**	**25 631 614,28**	**28 179 090,39**	**38 248 281,36**	**40 697 359,27**	**213 065 200,85**

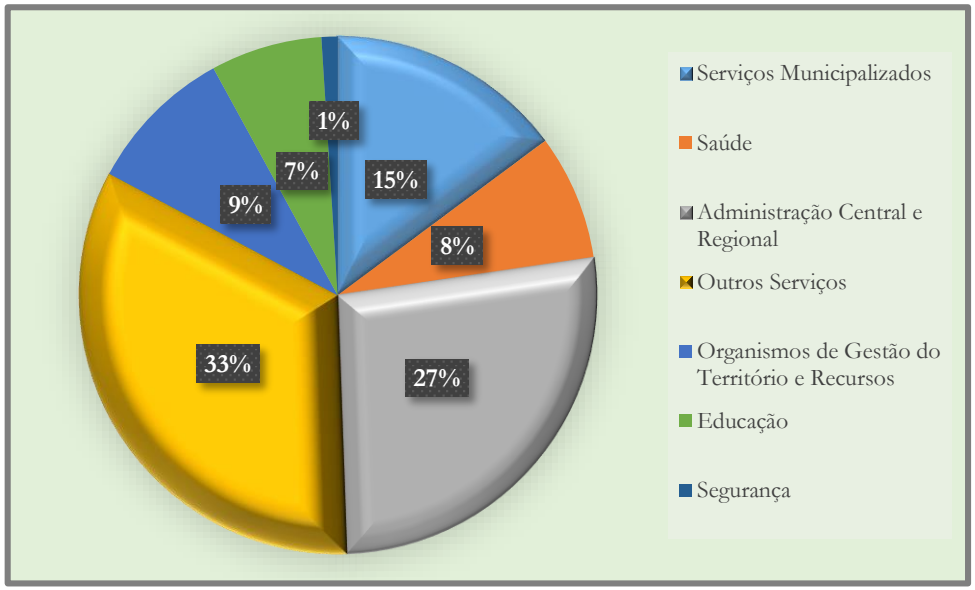


Figura 32 - Distribuição dos encargos de instituições públicas portuguesas com S4OS proprietário, por sector de actividade (2008-2015). Fonte: 'Base: contratos públicos online'.

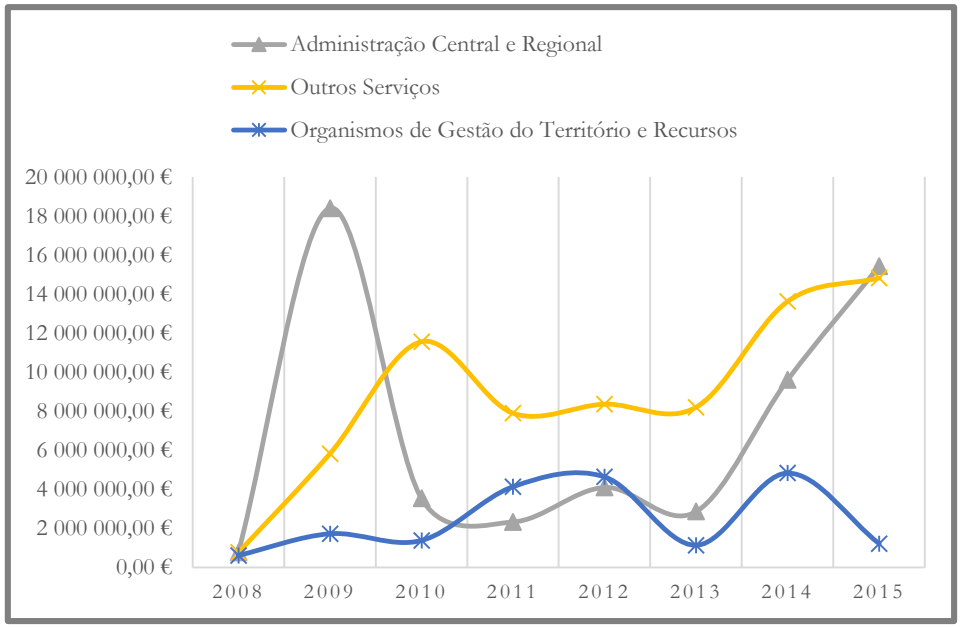
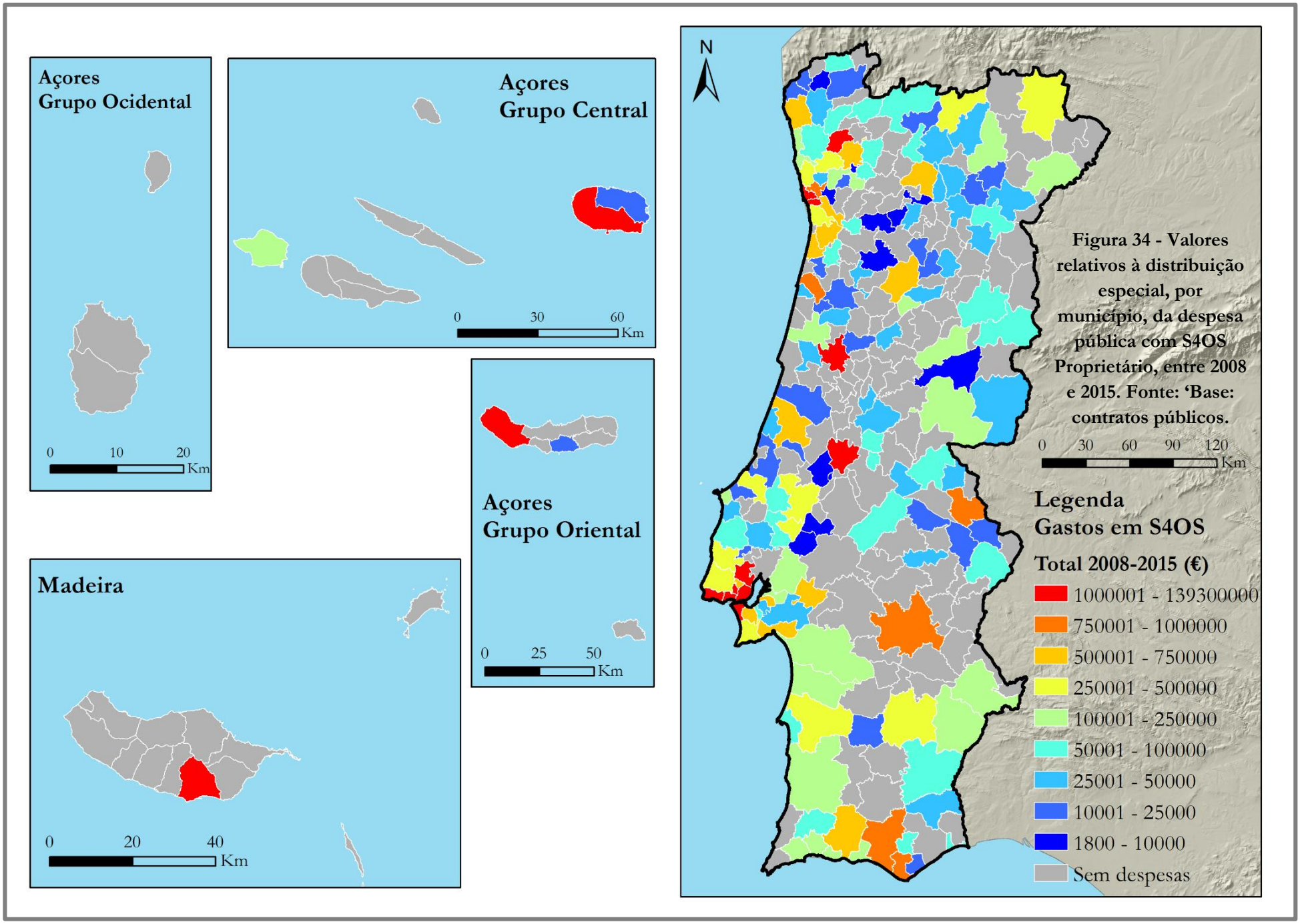


Figura 33 - Distribuição dos encargos de instituições públicas portuguesas com aquisição de S4OS entre 2008 e 2015, por sector de actividade e por ano. Fonte: 'Base: contratos públicos online'.

Por fim, fica um apontamento sobre a repartição espacial dos valores de despesa, relativos ao S4OS (Figura 34), que, grosso modo, à semelhança do que se constatou relativamente à espacialização dos custos com aquisição/renovação de licenças de *software* SIG, expressa uma polarização nas áreas metropolitanas de Lisboa (onde se destacam principalmente o concelho de Lisboa, mas também os de Oeiras, Loures, Amadora, Almada e Cascais) e do Porto, muito embora, tal como aconteceu anteriormente, exista uma assinalável diferença entre os dois concelhos polarizadores: Lisboa – 139 300 000 €, e Porto – 6 089 450 €. O já referido efeito de "litoralização", neste caso, diz respeito a uma região mais alargada uma vez que se incluem alguns municípios do litoral alentejano com valores bastante significativos, e cujo melhor exemplo é Santiago do Cacém. Próximo ao litoral, realce para Coimbra (1 990 965 €) e Braga (1 442 628 €). No interior do território continental, de uma forma geral, há diversos municípios com valores avultados, quase sempre bastante superiores, até, aos registados para o *software* SIG.

Na mesma linha do apontado no estudo desenvolvido por Patriarca *et al.* (2014a), os valores de que há registo para as regiões insulares revelam que, no arquipélago da Madeira apenas as instituições sediadas no concelho do Funchal usufruem, de forma representativa, dos serviços dos fornecedores por nós considerados, tendo despendido um valor total de 3 805 642 € no período considerado. No arquipélago dos Açores, Ponta Delgada e Angra do Heroísmo com 9 604 912 € e 3 209 946 €, respectivamente, também se juntam ao grupo dos concelhos com valores acima do 1 000 000 €.


Açores
Grupo Ocidental
0 10 20
Km
Açores
Grupo Central
0 30 60
Km
Açores
Grupo Oriental
0 25 50
Km
Madeira
0 20 40
Km
N
Figura 34 - Valores relativos à distribuição especial, por município, da despesa pública com S4OS Proprietário, entre 2008 e 2015. Fonte: 'Base: contratos públicos.
0 30 60 90 120
Km
Legenda
Gastos em S4OS
Total 2008-2015 (€)
1000001 - 139300000
750001 - 1000000
500001 - 750000
250001 - 500000
100001 - 250000
50001 - 100000
25001 - 50000
10001 - 25000
1800 - 10000
Sem despesas

4.3. Custos comparados entre soluções de SL/CA e SP

Fizemos notar já, por diversas vezes, que não existem alternativas sem custos, uma incontestável realidade que documentamos, por exemplo, na Tabela XVIII, que elenca os custos que as organizações da APP tiveram com a contratualização de serviços relacionados com o suporte/implementação de soluções baseadas em tecnologia livre e/ou de código aberto – total de 13 061 244,96 € (SIG – 4 475 053,24 €, e S4OS – 8 586 191,72 €). Começamos por destacar as abismais diferenças entre os custos com aquisição/renovação de licenças de SP com os associados ao SL/CA. No entanto, esta constatação não é suficiente para dizer que uma alternativa envolve mais custos que a outra, visto que, embora o período em apreciação seja o mesmo, a diferença entre o número de contratos adstritos a cada tipologia tecnológica é também muito grande, tal como se descreveu inicialmente, pois todos os dados apresentados até aqui não nos indicam a solução mais cara, apontando, antes, a mais utilizada.

Segundo a letra do OE 2013, apenas se poderão consumar despesas com aquisição de licenças de SP se for fundamentadamente demonstrada a inexistência de alternativas viáveis em SL/CA ou se o custo total deste último for superior à solução em proprietário. Assim sendo, o que pretendemos com este pequeno exercício é, a partir dos dados disponibilizados na plataforma governamental 'base', averiguar qual a solução que implica despesas mais avultadas. Alertamos, no entanto, para o facto de que qualquer ensaio deste tipo deve ser sempre relativizado, devendo somente motivar uma reflexão que contraponha os custos (em euros) que a APP registou ao longo dos últimos anos. Isto porque, pela complexidade deste fenómeno, os custos com suporte ou implementação de SL/CA dependem sempre dos objectivos da organização e características da tecnologia, ao contrário do SP, em que o preço contratual varia sobretudo com o número de licenças adquiridas.

Tabela XVIII - Despesas de entidades públicas com suporte/implementação de *Software* Livre e/ou de Código Aberto (SIG e S4OS) em Portugal, por ano e por empresa prestadora de serviços. Fonte: 'Base: contratos públicos online'

	Organismo/ Instituição	Despesas (€)								
		2008	2009	2010	2011	2012	2013	2014	2015	Total
SIG	Empresa1	0,00	0,00	0,00	0,00	17840,00	0,00	0,00	0,00	**17 840,00**
	Empresa2	0,00	314 830,84	766 251,37	814 575,70	421 730,52	87 067,40	1 679 972,63	372 784,78	**4 457 213,24**
	Total (SIG)	**0,00**	**314 830,84**	**766 251,37**	**814 575,70**	**439 570,52**	**87 067,40**	**1 679 972,63**	**372 784,78**	**4 475 053,24**
S4OS	Empresa3	7 187,50	183 440,00	68 588,00	325 472,75	173 085,00	131 605,00	240 140,00	459 240,60	**1 588 758,85**
	Empresa4	5 199,24	198 513,00	194 515,00	77 290,00	56 632,40	274 972,50	5 525,00	94 450,00	**907 097,14**
	Empresa5	0,00	0,00	0,00	0,00	0,00	0,00	19 024,00	0,00	**19 024,00**
	Empresa6	0,00	0,00	0,00	0,00	0,00	0,00	89 430,00	46 900,00	**136 330,00**
	Empresa7	0,00	0,00	0,00	0,00	0,00	37 840,00	22 160,00	0,00	**60 000,00**
	Empresa8	0,00	0,00	16 400,00	0,00	30 000,00	12 825,00	10 998,00	86 220,00	**156 443,00**
	Empresa9	0,00	0,00	15 555,00	0,00	34 600,00	35 300,00	0,00	0,00	**85 455,00**
	Empresa10	0,00	0,00	6 863,12	21 671,50	91 862,13	53 903,75	418 985,09	127 427,00	**720 712,59**
	Empresa11	9 300,00	122 370,01	0,00	161 000,00	200,00	124 490,00	27 550,00	0,00	**444 910,01**
	Empresa12	0,00	335 487,24	352 910,02	132 471,50	198 820,50	133 307,50	348 316,00	37 043,58	**1 538 356,34**
	Empresa13	0,00	131 739,60	315 515,35	142 195,00	118 092,50	8 127,70	78 531,70	174 608,80	**968 810,65**
	Empresa14	0,00	0,00	0,00	0,00	25 000,00	130 189,71	137 864,75	152 395,67	**445 450,13**
	Empresa15	32 315,76	99 934,00	281 399,00	56 950,00	479 625,21	59 500,00	0,00	0,00	**1 009 723,97**
	Empresa16	0,00	0,00	0,00	0,00	0,00	0,00	8 824,97	0,00	**8 824,97**
	Empresa17	0,00	21 770,00	0,00	0,00	71 250,00	0,00	139 601,00	0,00	**232 621,00**
	Empresa18	0,00	0,00	0,00	9 000,00	40 000,00	8 737,62	0,00	0,00	**57 737,62**
	Empresa19	0,00	98 189,45	2 000,00	36 900,00	22 800,00	36 232,00	9 815,00	0,00	**205 936,45**
	Total S4OS	**54 002,50**	**1 191 443,30**	**1 253 745,49**	**962 950,75**	**1 341 967,74**	**1 047 030,78**	**1 556 765,51**	**1 178 285,65**	**8 586 191,72**
	SIG+S4OS	**54 002,50**	**1 506 274,14**	**2 019 996,86**	**1 777 526,45**	**1 781 538,26**	**1 134 098,18**	**3 236 738,14**	**1 551 070,43**	**13 061 244,96**

Uma vez que os dois modelos de desenvolvimento de *software* representa volumes de negócio completamente díspares (consequência do claro domínio do SP no mercado), a comparação directa entre os valores das tabelas XIII e XVIII parece-nos pouco avisada. Posto isto, optámos por conceber e implementar uma estratégia que viabiliza, de forma prudente, esta comparação e que já foi explicada aquando da definição dos aspectos matriciais da metodologia utilizada nesta dissertação.

O Programa 5[108] foi construído de modo a permitir a comparação entre o total de despesa com SL/CA e o valor médio de despesa do somatório dos preços contratuais associados um conjunto de amostras com dimensão igual ao número de contratos associados a SL/CA.

Para uma primeira análise trabalhamos os contratos (SL/CA e SP) relativos ao *software* SIG com o Programa 5, resultando daí a informação da Tabela XIX, que apresenta os valores médios, mínimos e máximos entre um número variável de amostras constituídas por 99 contratos escolhidos aleatoriamente pelo algoritmo a partir da tabela que armazena as despesas com aquisição de SP SIG.

Tabela XIX - Diferença entre os gastos com *software* SIG Proprietário e os gastos com *Software* SIG Livre e/ou de Código Aberto. Valores calculados pelo algoritmo do Programa 5

N.º de Amostras	Valor médio	Mínimo	Máximo	Desvio-padrão	Diferença (SL/CA–SP)
10	4 088 332,46 €	2 622 360,06 €	10 405 881,90 €	2 290 665,23 €	386 720,78 €
100	4 562 086,09 €	2 726 965,00 €	10 967 492,91 €	2 232 635,06 €	87 032,85 €
1000	4 187 149,84 €	2 146 417,90 €	10 897 439,42 €	1 929 067,08 €	287 903,40 €
10000	4 237 542,04 €	2 051 912,65 €	11 960 133,71 €	2 052 866,26 €	237 511,20 €
100000	4 226 877,03 €	1 913 929,99 €	12 603 645,82 €	2 045 949,03 €	248 176,21 €
1000000	4 240 573,26 €	1 723 608,31 €	12 922 668,03 €	2 058 910,85 €	234 479,98 €
Desvio-padrão (coluna)	159 975,68 €	397 398,94 €	1 019 990,66 €	134 147,52 €	

Para o período sub-júdice, em média, e de acordo com a Tabela XIX, espera-se que uma amostra de 99 contratos para aquisição de SP SIG represente uma despesa entre os 4 000 000 € e os 4 500 000 €. Se assim for, independentemente do número de amostras usadas para calcular este valor, o SL/CA SIG é, em média, mais dispendioso que a alternativa proprietária, com uma diferença entre as duas e três centenas de milhar.

[108] **Ver Anexo A, p. 4.**

Apesar da amplitude de aproximadamente 8 000 000 €, o desvio-padrão mostra que os somatórios das várias amostras, que constituem a variável usada para calcular este indicador de dispersão, apresentam uma variação relativamente à média de cerca de 2 000 000 €, portanto, a variabilidade destes somatórios indica que, mesmo com o aumento do número de amostras, elas continuam a ser relativamente homogéneas.

A similaridade das amostras está também patente na reduzida variabilidade dos valores médios obtidos, ou seja, o programa não consegue gerar amostras de tal modo diversas que desvirtuem a média final. Assim, as amostras cujo total constitui o mínimo e o máximo entre elas dizem respeito a situações extremas, em que se reúnem, de um lado, os contratos com os valores mais baixos, e do outro, os contratos exageradamente mais caros, que acontecem em situações muito particulares e esporádicas.

Adiante, os mesmos indicadores, calculados para o S4OS (Tabela XX), mostram a elevada diferença entre o valor médio gasto em 320 contratos com aquisição de S4OS. Ao contrário do *software* SIG, mantendo-se as tendências apresentadas anteriormente, independentemente de considerarmos os valores médios, os mínimos ou os máximos, 320 contratos de SP envolvem sempre custos muito mais avultados que o SL/CA.

Tabela XX - Diferença entre os gastos com S4OS Proprietário e os gastos com S4OS Livre e/ou de Código Aberto. Valores calculados pelo algoritmo do Programa 5

N.º de Amostras	Valor médio	Mínimo	Máximo	Desvio-padrão	Diferença (SL/CA–SP)
10	32 087 595,31 €	23 254 491,23 €	43 407 790,59 €	7 732 602,25 €	23 501 403,59 €
100	29 920 730,11 €	16 619 688,28 €	44 831 149,55 €	7 117 024,73 €	21 334 538,39 €
1000	29 744 840,65 €	14 847 741,67 €	57 187 551,47 €	6 929 886,46 €	21 158 648,93 €
10000	29 622 830,38 €	14 152 190,13 €	62 182 456,88 €	6 924 713,08 €	21 036 638,66 €
100000	29 755 764,10 €	13 184 169,28 €	65 131 285,88 €	6 931 964,27 €	21 169 572,38 €
1000000	29 745 596,80 €	12 182 514,81 €	71 966 817,26 €	6 929 962,50 €	21 159 405,08 €
Desvio-padrão (coluna)	955 801,30 €	3 993 496,03 €	11 389 681,18 €	321 588,99 €	

À guisa de remate conclusivo deste tópico, o presente exercício constitui um modo expedito para calcular um valor médio para o custo de um determinado número de contratos, tendo em vista a comparação entre os custos com SP e os custos com SL/CA. Este método tenta contrariar a variação das verbas de contrato para contrato e a grande variedade de amostras com a dimensão pretendida que se podiam criar a partir do número total de contratos, calculando um indicador central gerado a partir da maior diversidade amostral possível. Não é uma metodologia perfeita,

estamos disso conscientes, mas é legítima e coerente, como o é a maior parte das que são desenvolvidas para finalidades análogas, pois nunca conseguiríamos mais do que um número médio para comparação, com as fragilidades que lhe são inerentes.

Para além de quaisquer enfoques que apontem outras limitações, dizer que esta é uma análise global que tem como único propósito motivar a reflexão sobre estes indicadores globais que nos permite comprovar que o SL/CA não é sempre menos dispendioso. Face a isto, mantemos a ideia de que cada caso é um caso, e em alguns deles o SL/CA pode ser mais rentável, noutros o seu contrário.

Este exercício denuncia a possibilidade de o SP SIG poder ser ligeiramente mais rentável que o SL/CA. Ainda assim, uma análise séria sobre este assunto merece outras considerações que divulgam novamente a importância que uma avaliação rigorosa tem no processo de selecção de uma das duas hipóteses (SP e SL/CA). A constatação de que o SP pode ser mais barato que o SL/CA depende dos objectivos e contexto subjacentes à sua adopção, pois, antes de a tomar como certa, importa apurar se o organismo está em condições de adoptar um *software* gratuito sem ter de recorrer a formação e suporte, se esta condição for garantida os custos iniciais seriam drasticamente reduzidos, tornando irrelevante o facto de o custo com suporte poder ser mais avultado ou não. Este processo é delicado e implica o reconhecimento das consequências que a adopção teria na estrutura organizacional, mormente na produtividade dos funcionários, pois a sua rejeição ou insatisfação relativamente à solução encontrada pelos administradores acabaria por trazer outras consequências, acarretando certamente incumprimentos e quaisquer custos que deles possam advir.

Salientamos ainda que, as diferenças entre SP e SL/CA SIG não são significativamente importantes, na medida em que os custos com SL/CA são elevados somente no início – esta é uma alternativa que a médio/longo prazo deixará de implicar custos, desde que o seu período de vida seja igualmente longo, isto porque não obriga renovação de licenças. Assim, apesar de os resultados parecerem, à primeira vista, desfavoráveis para o SL/CA SIG, conclui-se que não o são propriamente, uma vez que nos parece mais razoável optar por gastar 4 000 000 € durante oito anos, sabendo que nos próximos oito não haverá duplicação de custos (no caso de a solução ter sido implementada com sucesso, mostrando-se viável e fiável), do que gastar o mesmo, sabendo que nos oito anos seguintes se gastará outro tanto.

Sobre o S4OS, concluímos dizendo apenas que perante a qualidade mais do que documentada de *software* de escritório, estatística e gestão de informação (algo que ainda não se pode dizer com tanta certeza sobre o *software* SIG), estas despesas parecem-nos totalmente desprovidas de razão.

2ª Parte –

Benchmarking: testes de desempenho comparado entre soluções de SP e SL/CA em ambiente SIG – exemplos de aplicação em contextos específicos da Administração Pública Portuguesa

CAPÍTULO I — SELECCIONAR E TESTAR: BASES CONCEPTUAIS SOBRE *SOFTWARE TESTING*, E ENQUADRAMENTO TEÓRICO DE MÉTODOS PARA HIERARQUIZAÇÃO DAS ALTERNATIVAS A SEREM EXPLORADAS E TESTADAS

Quando confrontados com a imposição ou necessidade de adoptar SL/CA na sua organização, **os** administradores públicos sentem-se desorientados (Marsan & Paré, 2013), nomeadamente porque uma das adversidades que se colocam à implementação de SL/CA em qualquer instituição tem a ver com o reduzido nível de confiança e de conhecimento que os administradores têm relativamente a ele.

Neste ponto do trabalho, tendo em conta os tópicos enunciados na 1ª Parte deste documento que impelem a APP a justificar as suas opções no que diz respeito à aquisição de *software* e a considerar necessariamente alternativas livre e/ou de código aberto, interessa-nos, entre os conjuntos de factores/variáveis a incluir no processo de avaliação comparada de dois recursos SIG, a averiguação da qualidade e a sistematização das vantagens técnicas dos programas em análise, sendo este um indicador determinante, com elevado peso no momento da decisão (Glynn *et al.*, 2005; Fitzgerald, 2011; OSEPA, 2012h; Bouras *et al.*, 2013 Sarrab & Rehman, 2014).

No concurso com o SP, o SL/CA tem de respeitar padrões de qualidade exigíveis por potenciais utilizadores, sob pena de estes virem a preferir alternativas que, embora sejam, à partida, mais dispendiosas, oferecem mais garantias de eficiência, acabando por se mostrar economicamente mais vantajosas. Parece ser pacífica a ideia de que qualquer administrador consciente, no balanço das várias possibilidades de que dispõe, testará o *software*, atestando a sua qualidade, eficácia, eficiência, fiabilidade e credibilidade, na medida em que não é sensato escolher um *software* SIG sem que antes se formulem este tipo de ensaios e considerações. Indo mais além, diríamos até que qualquer processo de adopção poderia ser inviabilizado se a alternativa candidata a eleição não funcionar como o esperado, fazendo com que este domínio tenha mais peso que os restantes.

O capítulo que agora se inicia materializa justamente uma reflexão sobre os tipos de avaliação técnica que se podem realizar, bem como sobre o papel que o processo de avaliação das capacidades do *software* deverá ter na tomada de decisão sobre a adopção de um determinada programa ou sistema informático.

É do senso comum pensar-se que a estipulação de testes que tentam averiguar a robustez das ferramentas de um *software* SIG poderá ser uma tarefa trivial. Puro engano, mesmo para aqueles que, sendo conhecedores do assunto possam assim entender este processo que é deveras complexo. Este tem de ter em conta os princípios terminológicos relacionados com a Engenharia dos Requisitos quando associada a processos de *Software Testing*. Mas, não só. Normalmente, quando empreendemos esforços que visam a comparação de *software* tendo em conta a concretização de objectivos específicos, constatamos que existe uma grande variedade de soluções com potencial. Ora, na prática, tal constitui um problema, visto que, na maior parte das situações, não há tempo suficiente para as testar todas, sendo necessário circunscrever o universo de alternativas, testando efectivamente aquelas que, depois de uma análise preliminar, apresentam evidências de satisfazerem com maior comodidade, rapidez e fiabilidade, os nossos objectivos.

Esta análise preliminar parte de uma lista mais ou menos extensa de hipóteses e alternativas, hierarquizadas segundo um conjunto de métodos e critérios que importava inventariar, estudar e definir previamente ao ensaio de *Software Testing*.

Neste contexto, o capítulo que agora se inicia apresenta a seguinte "ordem de trabalhos":

1. **Critérios para avaliação da qualidade do *software* e a desconhecida arte do *Software Testing: Black-box vs White-box testing***
    i. Esclarecer conceitos e técnicas relacionadas com o *Software Testing*.
    ii. Identificar trabalhos cujo principal propósito tenha sido a comparação de parâmetros que representem a eficiência e grau de desempenho de diversos *software* SIG (proprietários e livres).
2. **Análise Multicritério e Processo Analítico de Hierarquização enquanto mecanismos de apoio à decisão**
    i. Definir genericamente os conceitos estruturantes destas duas metodologias de apoio à decisão.
    ii. Esclarecer o modo como estas duas técnicas integram o trabalho, permitindo-nos seleccionar os *software* que serão explorados e testados nos capítulos subsequentes.

I. Critérios para avaliação da qualidade do *software* e a desconhecida arte do *Software Testing: Black-box vs White-box testing*

> *"Existe uma regra bem conhecida que indica que, num projecto normal de construção de um programa informático, aproximadamente 50% do tempo despendido e mais de 50% do custo total de produção são consumidos em tarefas que visam o teste do programa ou do sistema a ser desenvolvido"*
>
> Glenford Myers; Tom Badgett & Corey Sandle (2012:ix)[109]

Bouras *et al.* (2013) assumem que, na adopção de qualquer SL/CA, deve ser ponderado um conjunto de parâmetros directamente relacionados com as características da tecnologia, entre os quais constam os seguintes:

i. Funcionalidade;
ii. Suporte;
iii. Manutenção;
iv. Longevidade;
v. Segurança;
vi. Usabilidade (*user friendly*)[110];
vii. Interoperabilidade;
viii. Flexibilidade e grau de customização;
ix. E performance.

[109] No original, *"It was a well-know rule of thumb that in a typical programming project approximately 50 percent of the elapsed time and more than 50 percent of the total cost were expended in testing the program or system being developed". In* MYERS, Glenford; BADGETT, Tom & SANDLE, Corey (2012) – **The art of *Software* Testing**. John Wiley & Sons, 3ª Edição, Nova Jersey, 240 p.

[110] A usabilidade é o campo da engenharia de *software* que se preocupa com a interacção do ser humano com o computador, tendo como objectivo específico tornar as interfaces de comunicação entre o homem e a máquina altamente funcionais, intuitivas e amigas do utilizador, de modo a maximizar a operabilidade, minimizando tempos despendidos na adaptação a uma interface onde o operador sente dificuldades em se movimentar (Akbari & Rajabi, 2013).

Na mesma linha, Sarrab & Rehman (2014) descrevem e implementam um modelo que agrega um conjunto de critérios bastante semelhantes aos anteriores, e que visam a escolha da melhor solução (Figura 35). Estes autores centram-se na avaliação de diferentes aplicações para gestão de redes informáticas, recorrendo, para o efeito, a este modelo, que descreve as características das soluções em três domínios: i) qualidade do sistema; ii) qualidade da informação e, iii) qualidade do serviço (Figura 35). Para cada um dos *software* em exame, a cada parâmetro associado a cada dimensão atribuiu-se um *score* compreendido entre 1 e 5 sendo: 1 – menos importante; 3 – neutro; 5 – muito importante. Este *ranking* valorativo quantificável, é acompanhado por um comentário que dá a conhecer factos importantes sobre aquela solução e suas características.

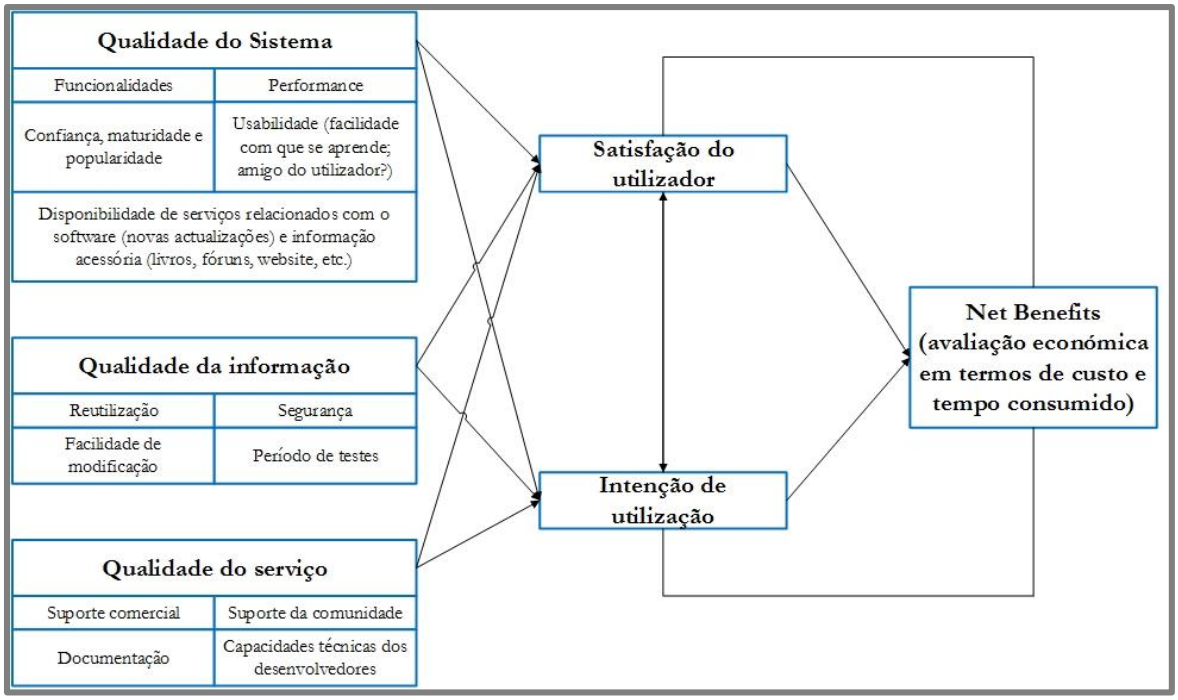


Figura 35 - Modelo de sucesso para uma determinada solução informática. Fonte: Sarrab & Rehman (2014), p. 2.

Estamos perante um modelo completo que intenta avaliar toda a componente tecnológica do *software*, incluindo também outros critérios directamente relacionados com ela. A apreciação da dimensão financeira é relegada para uma fase posterior. Cremos que, em primeiro lugar, no processo de deliberação sobre a aquisição, ou não, de determinada hipótese, tem de estar o escrutínio das propriedades técnicas do *software*, só depois se devem ponderar os elementos financeiros. Esta parece-nos ser a opção lógica, pois mesmo que o custo de licenciamento (em euros) de um produto seja zero, se a qualidade do sistema, da informação e do serviço não for satisfatória, o custo verdadeiro poderá superar o de outra alternativa que envolva despesas com

licenciamento. Para além disto, o custo financeiro só será conhecido depois de se saber que tipo de suporte e de formação se tem de contratar, e essa constatação só será clara quando tiverem sido reunidas as informações sobre os critérios definidos neste esquema. Logo, a avaliação financeira deve estar *a jusante* dos métodos e técnicas de medição da qualidade do *software*. Se a tecnologia não corresponder às especificações, essa alternativa tem de ser simplesmente eliminada do processo de tomada de decisão!

Embora tenhamos ao nosso dispor um exemplo tipo daquilo que o processo de avaliação do *software* tem de ser, no âmbito desta dissertação, não nos é possível avaliar todos os parâmetros incluídos neste esquema, nomeadamente os relacionados com a qualidade de informação. De acordo com Sarrab & Rehman (2014), esta dimensão remete para as características do código-fonte que seriam desejáveis, a saber:

i. A estrutura do código deve permitir uma leitura acessível, de modo a facilitar a percepção de como o programa foi estruturado, assim, quanto mais intuitiva for a leitura, maior é a facilidade de adaptação do código às necessidades do utilizador;

ii. O programa deve estar construído de forma a facilitar a reutilização do seu código, possibilitando a integração de novas funcionalidades.

iii. O *software* deve estar preparado para simplificar tarefas que procuram detectar limitações e defeitos existentes no código-fonte (optimização do *software testing*).

iv. Por fim, o programa deve garantir a confidencialidade, segurança e integridade da informação que processa, minimizando as vulnerabilidades – aqui, integridade aparece relacionada com a disponibilidade de mecanismos de controlo que asseguram a integridade do sistema.

A exploração destes aspectos é absolutamente relevante, contudo, não dispomos de conhecimentos avançados de programação para análise da qualidade, consistência e integridade lógica do código-fonte. Assim sendo, focar-nos-emos nas outras duas dimensões.

Nos exercícios que apresentaremos de seguida, os aspectos relacionados com a qualidade do serviço serão, na medida do possível, contabilizados no processo que visa a escolha dos programas que serão efectivamente testados numa fase posterior, em que procuraremos percepcionar sobretudo a sua performance, as suas funcionalidades e as suas limitações.

A apreciação, em geral, da qualidade de um *software*, e, em particular, da sua performance encorajou-nos a estudar a epistemologia daquilo que a literatura da especialidade designa como *Software Testing*.

A engenharia de requisitos é o ramo da engenharia de *software* que envolve as actividades de eleição, análise, especificação, extracção, documentação e validação de um conjunto de requisitos a que o *software* deve responder (Unterkalmsteiner *et al.*, 2015). Para garantir que as exigências colocadas ao *software* são satisfeitas quando ele corre, recorre-se a outra disciplina com uma forte relação com a primeira. O *software testing* é um conceito definido como o processo ou série de processos desenhados para garantir "que o código faz aquilo para que foi criado" (recorde-se que "aquilo" está expresso nos requisitos), e também para nos certificarmos de que não faz algo não intencional (Myers *et al.*, 2012; Unterkalmsteiner *et al.*, 2015).

De acordo com Unterkalmsteiner *et al.* (2015) o desenvolvimento de um *software* é sempre um esforço colaborativo de especialistas, cada um contribuindo para parte da solução – a experiência, inteligência e capacidades são distribuídas por diferentes papéis, o que implica que tenha de haver um excelente trabalho de coordenação entre as pessoas envolvidas no projecto em fases essenciais, caso contrário, o projecto está condenado a falhar. Portanto, o desenvolvimento de *software* consiste num conjunto de transições entre a seguinte sequência de etapas:

i. Conceito do sistema;
ii. Especificação dos requisitos;
iii. Análise e *design*;
iv. Implementação;
v. Provação;
vi. E manutenção.

Estes preceitos são sistematizados no Modelo-V (Figura 36) do desenvolvimento de *software*, originário da engenharia de sistemas e adoptado pela engenharia de *software*. Este modelo ilustra transições verticais entre as actividades típicas do desenvolvimento de *software*, às quais estão inevitavelmente associadas actividades que têm como propósito testá-lo, assegurando a qualidade dos produtos resultantes. Por isso se diz que 50% dos custos do desenvolvimento do *software* referem-se a actividades de teste, despesas que, segundo Afzal *et al.* (2016), continuarão a aumentar com o crescimento da complexidade dos *software*.

Testar é fundamental! Tudo o que se disse demonstra como o *Software Testing* é importante na construção de qualquer programa de computador, todavia, quando se trata de SL/CA, estas técnicas de avaliação de qualidade não devem ser reservadas à produção e construção do *software*, pelo contrário, devem ser implementadas recursivamente sempre que pretendemos adoptar este

tipo de recurso numa estrutura organizacional. Como se sabe, o SL/CA é um fenómeno muito particular e complexo, no qual é difícil controlar modificações e actualizações, apontar quem desenvolveu (?), como e porque desenvolveu (?), e que testes executou (?) - Lerner & Tirole (2001), Fuggetta, (2003). No que diz respeito a este tipo de *software*, em alguns casos, não é evidente sequer se se confirmaram os requisitos no momento da sua produção. Nesta medida, nunca será demais testar SL/CA, até porque serve de catalisador na reunião de sinergias em torno da comunidade para correcção de erros que se possam vir a descobrir (*bugtracking*).

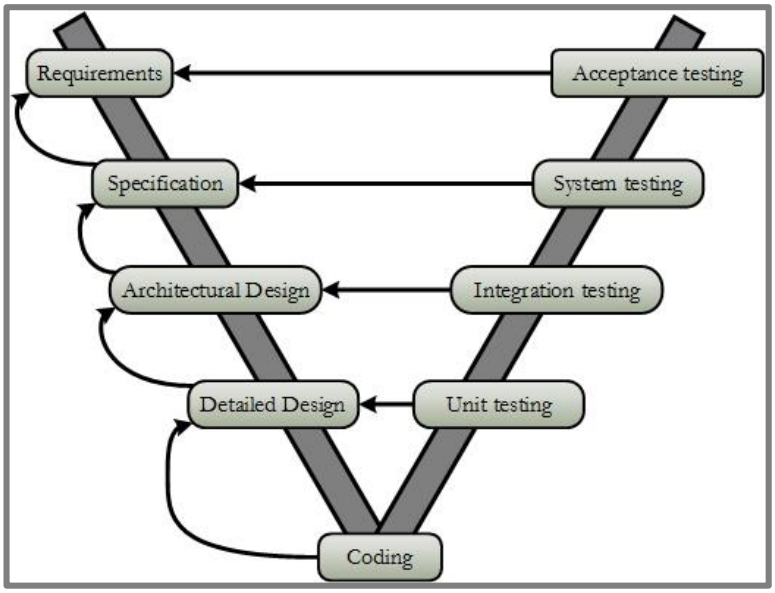


Figura 36 - Modelo-V do desenvolvimento de *software*. Fonte: Unterkalmsteiner *et al.* (2015), p. 63.

Apresentado o motivo que nos leva a testar SL/CA, resta-nos esclarecer os modos de o fazer. Em termos gerais, existem duas estratégias predominantes e complementares: o *Black-Box Testing* e o *White-Box Testing* (Myers *et al.*, 2012; Kempka *et al.*, 2015).

A primeira modalidade, também conhecida como *Data-Driven* ou *Input/Output-Driven Testing*, é um método que "vê" o programa como uma caixa negra, ou seja, o operador que realiza o teste ignora completamente o comportamento interno e a estrutura do programa, concentrando-se em encontrar circunstâncias em que o programa não se comportou de acordo com as especificações (Myers *et al.*, 2012). Trata-se de uma técnica que se baseia somente nas especificações, sem tirar partido das vantagens trazidas pelo conhecimento da estrutura interna do código. É um processo sobretudo empírico, individualista e intuitivo, no qual se testam cenários com interesse para o operador, expondo-se facilmente problemas (Hamlet, 2015). No entanto, com este método é

impossível proceder-se a uma avaliação exaustiva do programa. De acordo com Myers *et al.* (2012), se quiséssemos levantar todos os erros do programa, teríamos de usar todas as entradas possíveis (*Exhaustive Input Testing*), e claro, para alguns programas, as possibilidades de *input* seriam virtualmente infinitas. Exemplificando, considere-se o código do Programa 13[111], o qual, a partir de três valores inteiros que correspondem ao comprimento dos três lados de um triângulo, nos indica se essa figura seria um triângulo equilátero, isósceles ou escaleno; repare-se que se incluiu, por lapso, uma condição que declara que se os três lados tiverem o valor 3842, o triângulo será considerado como escaleno; ignorando o código, só nos aperceberemos deste erro se introduzirmos todas as combinações possíveis de lados de um triângulo.

Por seu turno, a estratégia *white-box*, ao contrário da primeira, privilegia o exame da estrutura do programa, extraindo informações relativas à sua coerência lógica, consequentemente negligencia a relação entre o código e as suas especificações. Segundo Myers *et al.* (2012), o propósito deste método é estabelecer uma abordagem análoga ao *exhaustive input testing* da estratégia *black-box*, mas retirando partido de análises sobre o código-fonte. Esta estratégia é conhecida na literatura como *Exhaustive Path Testing* (Kempka *et al.,* 2015). De acordo com Myers *et al.* (2012), ela indica que, para testar completamente o programa, é necessário fazer com que ele execute, pelo menos uma vez, cada condição (*statment*). Todavia, isto suscita duas limitações relevantes que importa vincar.

Em primeiro lugar, se encararmos o algoritmo como um percurso, em que cada declaração *if*, *elif* ou *else* se desdobra em vários tramos (Figura 37), o número de caminhos lógicos existentes no programa seria astronomicamente elevado. A Figura 38 é um programa básico. O diagrama é um grafo de controlo de fluxo. Cada nó representa uma declaração de uma condição e o respectivo segmento de código subsequente que termina com uma nova declaração que obriga a transferência do controlo (representada aqui pelos arcos) para outro segmento de código. O diagrama descreve um programa com 10 a 20 declarações que consiste num ciclo que o impele a repetir o processo até 20 vezes. Dentro do corpo do ciclo encontra-se um emaranhado de declarações *if*. Assim, determinar o número de todos os caminhos lógicos é o mesmo que determinar o número total de percursos únicos que estabelecem o movimento entre o ponto a e o ponto b (assumindo, claro está, que todas as decisões no programa são independentes umas das outras). Este número é aproximadamente 10^{14}, e é consequência da seguinte soma: $5^{20} + 5^{19} + \cdots + 5^{1}$, onde 5 é o número de arcos que atravessam o corpo do ciclo (Myers *et al.*, 2012).

[111] **Ver Anexo A – p.A12.**

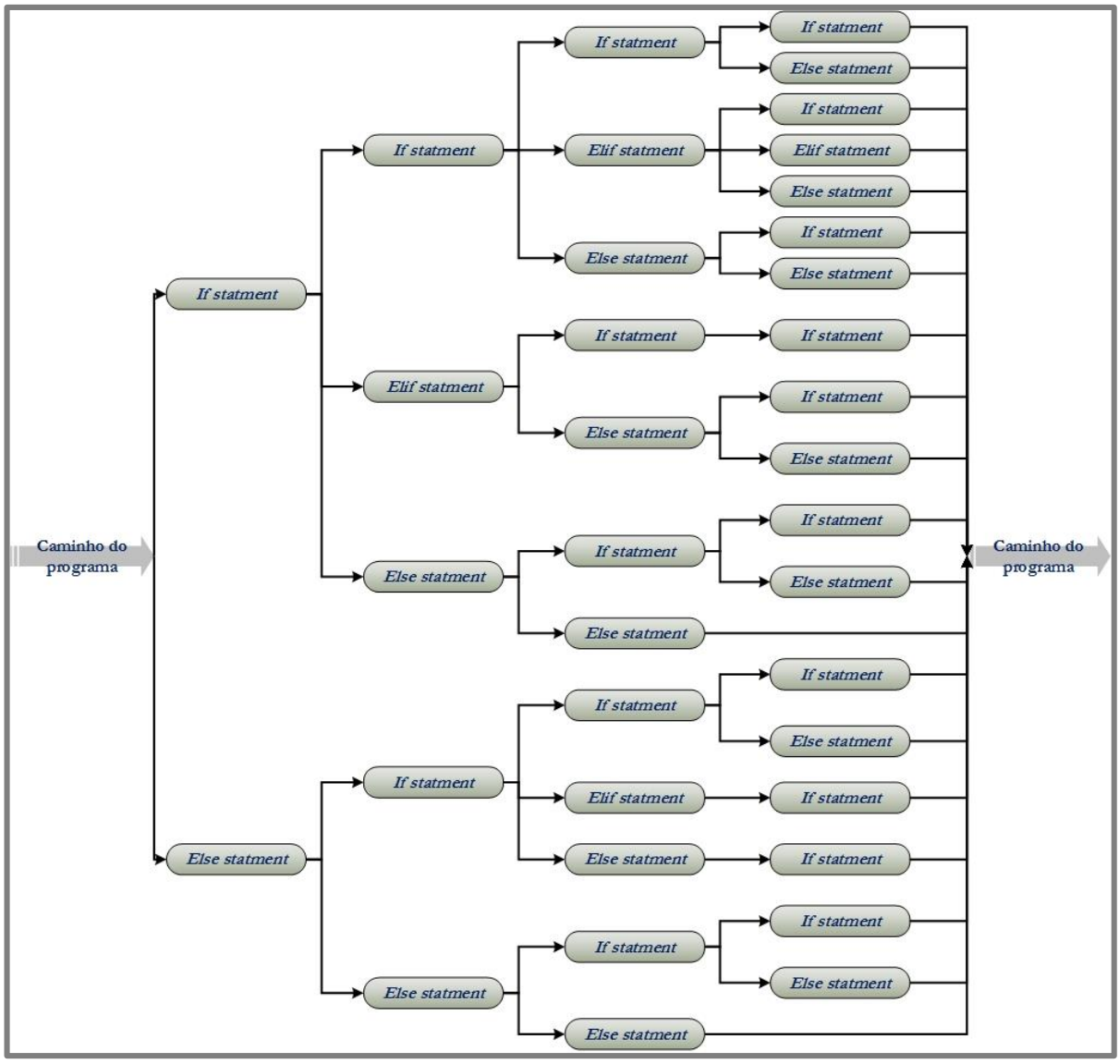


Figura 37 - Programa visto como uma encruzilhada de condições (*statment*) que multiplicam os caminhos possíveis desde o início até ao fim.

Verificando cada *test case* a cada cinco minutos, o operador levaria aproximadamente 1 bilião de anos para experimentar todos os caminhos. É certo que na maioria dos programas as decisões não são independentes de todas as outras, o que significa que o número de caminhos possíveis é menor, ainda assim, estes programas são muito mais complexos que o da Figura 38. Portanto, à semelhança do que acontece no *Exhaustive Input Testing*, verifica-se que *Exhaustive Path Testing* é pouco prático, se não mesmo impossível (Myers *et al.*, 2012).

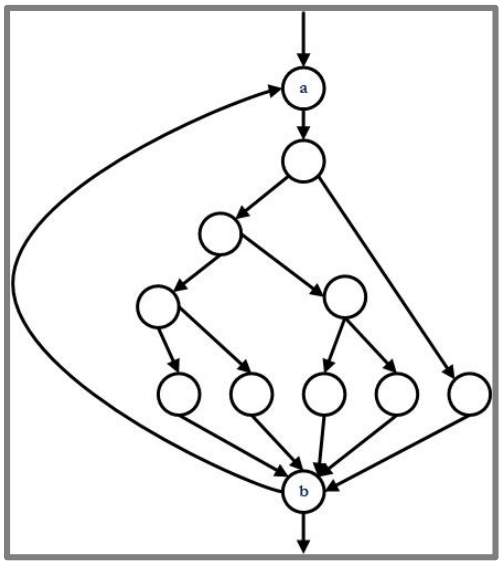


Figura 38 - Grafo de controlo de fluxo de um pequeno programa. Fonte: Myers *et al.* (2012), p. 11.

Em segundo lugar, mesmo que a abordagem do *Exhaustive Path Testing* signifique um teste completo em que todos os caminhos do programa são executados, não há garantias de que ele fique isento de erros. Myers *et al.* (2012) referem três explicações para isso:

i. Por um lado, esta técnica não garante que o programa respeita a sua especificação. Por exemplo, é pedido ao operador para escrever um algoritmo que organize um conjunto de números por ordem ascendente, mas, por lapso, ele cria um que organiza esses elementos por ordem descendente. Numa situação como esta, o *Exhaustive Path Testing* de nada serviria porque, embora a integridade lógica do código estivesse assegurada, o erro do programa não está propriamente no raciocínio que concretiza o código, mas sim no facto de ele não corresponder à especificação.

ii. Por outro lado, o programa pode estar incorrecto porque faltam caminhos, e, claro, este procedimento não detectará a sua ausência.

iii. Por fim, este método não cobre erros relacionados com a sensibilidade dos dados. Suponhamos que no programa pretendemos comparar dois números, sendo que um deles é a diferença entre outros dois (Programa 14[112]). Como se vê, a declaração *if* contém um erro porque deveria comparar o valor absoluto de a-b. A percepção deste problema está dependente dos valores atribuídos a a e b e poderá não ser detectada mesmo que se executem todos os caminhos do programa.

[112] **Ver Anexo A, p. A13.**

Concluindo, embora se considere que o método *white-box* seja superior ao *black-box*, nenhum dos dois é totalmente eficaz, o que faz deles complementares (Myers *et al*., 2012).

A presente revisão de literatura e conceitos não é exaustiva e desconsidera um largo conjunto de métodos utilizados no âmbito do *software testing*, alguns dos quais tentam ultrapassar as limitações do *exhaustive path testing* (Kempka *et al*., 2015), referidos, por exemplo, na revisão bibliográfica apresentada por Kanewala & Bieman (2014) ou Afzal *et al.* (2016). Ainda assim, esta breve incursão no *software testing* permite antecipar o tipo de orientação que pretendemos dar aos nossos exercícios que visam o teste de diferentes *software* SIG.

O ideal seria executar procedimentos do tipo *black-box* e *white-box*, visto que se tratam de mecanismos complementares. Porém, humildemente admitimos dificuldades em implementar exercícios que impliquem a interpretação da estrutura interna do código-fonte dos algoritmos a estudar, fruto da insipiência dos nossos conhecimentos em informática e computação. Para além disto, é do conhecimento comum que grande parte dos funcionários públicos que trabalham com SIG também não possuem estas competências. Por estes dois motivos, mas também porque o nosso objectivo é essencialmente encontrar limitações e deficiências na performance e funcionalidades dos *software* a serem testados, optámos por uma estratégia do tipo *black-box*.

Com esta avaliação desejámos confirmar o potencial que o SL/CA SIG tem para a modelação espacial e para os organismos da APP que têm obrigações nesta área, justificando a necessidade de ser criada uma plataforma que integre parâmetros relacionados com performance, funcionalidades e qualidade com outras dimensões, tendo em vista a selecção consciente e informada de SL/CA na APP, e acrescentando também conhecimento sobre as verdadeiras capacidades e limitações do *software* SIG do tipo livre e/ou de código aberto, pois segundo nos foi possível perceber, existem muito poucos exercícios que exploram estas questões.

Encontramos somente três exposições que comparam o desempenho de alguns projectos e contrapõem o SP ao SL/CA. A de Akbari & Rajabi (2013) é de todas a mais interessante, na medida em que foca aspectos relacionados com a performance, escalabilidade[113] e usabilidade de três *software* SIG (GRASS GIS 6.3.0, SAGA 2.0.0 e ILWIS 3.4) em diferentes tarefas – não se limita a comparar o desempenho.

Por sua vez, Sátiro & Simões (2013) executaram um conjunto de ferramentas que tinham como objectivo identificar as regiões potencialmente mais adequadas para a silvicultura na bacia do Rio

[113] Entenda-se escalabilidade como a capacidade de um sistema, uma rede ou um processo tem para suportar um aumento substancial do volume de dados sem que o seu desempenho piore ao ponto de pôr em causa a sua utilização (Akbari & Rajabi (2013).

Paraíba do Sul (Brasil), e compararam os resultados obtidos com o ArcGIS e com o gvSIG. Trata-se de um exemplo aplicado a um problema concreto, no entanto, em nosso entender, inválido porque não foi usado o mesmo modelo de dados nos dois momentos de aplicação da metodologia.

Por último, Patriarca *et al.* (2014a) testaram vários *software* SIG, entre os quais um SP, o QGIS e o gvSIG, chegando à conclusão que, em termos gerais, os SL/CA envolvidos apresentam um fraco desempenho no corte topológico da CRIF 2011 pelos limites da Região de Coimbra (formato vectorial e matricial), embora considerem que as diferenças de desempenho não foram significativas ao ponto de rejeitar, desde logo, as alternativas livres.

Estes ensaios servem sobretudo para nos lembrar que qualquer ensaio de *benchmarking* é sempre subjectivo e incompleto porque depende de vários factores, desde logo, das competências do operador, mas também do tipo de tarefas de *software testing* a serem aplicadas, bem como do tipo de processos de análise e de geoprocessamento a serem ensaiados, sem esquecer as condições do ensaio. Acima de tudo, servem também para vincar que não é possível detectar, analisar e interpretar exaustivamente todos os erros e limitações de um programa, como referem Myers *et al.* (2012, op. cit. p.41)[114].

Em síntese, na medida em que testar até à exaustão um *software* é impossível, é aceitável e válida a intenção de apresentar exercícios adequados às nossas habilitações (testes do tipo *black-box*), e contextualizados com as competências de determinados organismos da APP, dando, assim, o nosso pequeno contributo para uma melhor compreensão de como o SL/CA SIG pode ser usado nestes e noutros contextos semelhantes.

[114] *Testing, however creative and seemingly complete, cannot guarantee the absence of all errors… Test-case design is so important because complete testing is impossible.*

2. A Análise Multicritério e o Processo Analítico de Hierarquização enquanto mecanismos de apoio à decisão

> *"Quando nós pensamos, identificamos objectos ou ideias e as relações entre eles. Quando identificamos qualquer coisa, decompomos a complexidade da realidade que encontramos. Quando descobrimos relações, sintetizamos. Este é um processo fundamental subjacente à percepção: decomposição e sintetização. A elaboração deste conceito e as suas implicações práticas interessam-nos aqui."*
>
> Thomas Saaty (1988:3)[115]

Como se viu anteriormente, o universo das TIG encontra-se actualmente em constante expansão. Voltamos a fazer referência à Tabela IX (cfr. p. 54), na qual são apresentados inúmeros projectos que desenvolvem SL/CA para aplicação em áreas afins aos SIG, como DR, CAD e SGBD. Ainda que seja com agrado que assistimos a esta expansão, e tendo em conta que o nosso objectivo é testar e comparar *software*, a existência de um elevado número de alternativas levanta-nos vários problemas.

Mesmo circunscrevendo o número de soluções àquelas vocacionadas para a manipulação e processamento de dados geográficos, resta-nos um conjunto considerável de elementos. Não é humanamente possível, pelo seu número, considerá-los a todos num trabalho com estas características. Se bem que tal exercício estaria desprovido de razão visto que, a maior parte dos projectos de SL/CA SIG, ou possuem ferramentas muito específicas[116], ou disponibilizam somente as ferramentas mais simples e comuns num *software* SIG, exibindo um grau de maturidade significativamente baixo.

---

[115] No original, *"When we think, we identify objects or ideas and also relations among them. When we identify anything, we decompose the complexity which we encounter. When we discover relations, we synthesize. This is the fundamental process underlying perception: decomposition and synthesis. The elaboration of this concept and its practical implications interest us here". In* SAATY, Thomas (1988) – **The analytic hierarchy process: planning, priority setting, resource allocation**. Mcgraw-Hill, ISBN 0070543712.

[116] Por exemplo, *software* como o Kalypso ou o DIVA-GIS são especializados num tipo de modelação específico – o primeiro dispõe de ferramentas adequadas à modelação no âmbito dos riscos naturais, nomeadamente risco de inundação; o segundo reúne algoritmos construídos com o intuito de processar dados bioclimáticos.

Assim, antes de mais nada, devem-se procurar técnicas que nos permitam hierarquizar as opções SIG, aquelas que aparentemente parecem ser mais capazes de lidar e competir verdadeiramente com a concorrência do SP líder de mercado na área dos SIG, tendo em conta os tipos de exercícios que preparámos.

2.1. Métodos e técnicas de eleição para tomada de decisão: Análise Multicritério aplicada à selecção de *software* SIG

Não são muitos os trabalhos que nos esclarecem quanto aos métodos a usar para seleccionar *software* SIG. Neste âmbito, destacamos os trabalhos de Chen *et al.* (2010), Dobesova (2013) e Eldrandaly & Naguib (2013).

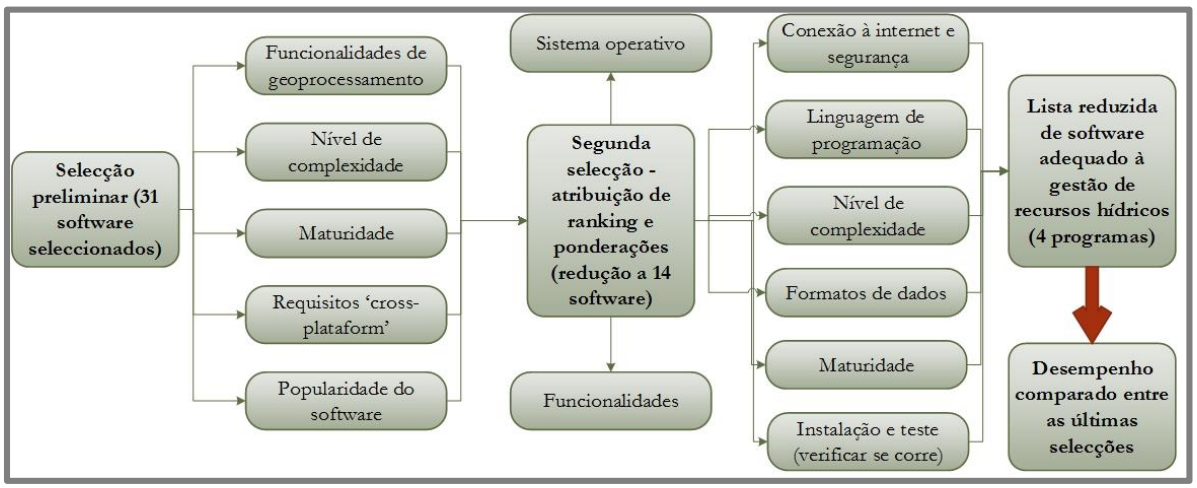


Figura 39 - Diagrama de fluxo com os processos e critérios utilizados para seleccionar o *software* com propriedades que garantem a gestão eficiente de recursos hídricos. Fonte: Chen *et al.* (2010), p. 256.

O primeiro dos trabalhos referidos sugere uma metodologia que, segundo os critérios apresentados no fluxo de trabalho da Figura 39, tem como finalidade seleccionar o melhor *software* SIG para gestão de recursos hídricos. Trata-se de um modelo de análise estritamente qualitativa, numa primeira fase, na qual se avaliam de modo subjectivo parâmetros como as funcionalidades do *software*, o nível de complexidade, a maturidade, a possibilidade de correrem em vários sistemas operativos e a popularidade. Esta fase, da qual resultaram os 14 *software* que melhor responderam a estes critérios, caracteriza-se pela subjectividade e relatividade; a primeira está presente porque a valoração qualitativa atribuída é fruto da opinião pessoal dos autores (advogam, por exemplo, que o GRASS apresenta um grande nível de complexidade, no entanto outros utilizadores podem não

ter a mesma opinião); a segunda (relatividade), surge na medida em que a direcção da valoração atribuída é relativa ao problema em que se está a trabalhar (melhor *software* para gestão de recursos hídricos) e à máquina que se está a utilizar (se o computador usado tivesse componentes de *hardware* mais possantes, um *software* com 1 GB não seria demasiado[117]).

Com a triagem de um grupo mais pequeno de *software*, os autores supramencionados incrementaram, na fase seguinte, um estudo quantitativo das 14 alternativas, em que os oito critérios considerados (sistema operativo, funcionalidades, conexão à internet e segurança, linguagem de programação, nível de complexidade, formatos de dados, maturidade e instalação) foram pesados (ponderados), atribuindo-se, a cada *software* um *ranking* (0 – pobre; 0,5 – médio; 1 – bom) em cada um dos critérios, para posteriormente se calcular o *score* final segundo a expressão:

$$\sum_{i=1}^{8} (C_i W_i S_i) \times 100$$

onde, C_i é o critério, W_i é o peso de critério específico e S_i é o factor de hierarquização (*rank*) de um critério particular.

Daqui construiu-se uma lista reduzida de *software* com maior potencialidade para corresponder às expectativas (com apenas quatro aplicações), que viram o seu desempenho testado em aspectos como o tempo de arranque, o tempo necessário para ler uma imagem *tif* de 125 Mb, o tempo que a interface precisa para restaurar a visualização depois de fazer *pan* sobre uma imagem *tif* de 125 Mb, e o tempo despendido para fazer *zoom*.

Embora não se refira nenhuma vez o termo Análise Multicritério (AMC), na prática, esta é uma metodologia que aplica os principais pressupostos da AMC.

Dobesova (2013), por sua vez, apresenta também um método para comparação de soluções SIG, o *CartoEvaluation*, baseado na AMC e no método *Goal-Question-Metric*, desenvolvido especificamente para a avaliação de *software* – este processo consiste na identificação dos principais requisitos que o programa tem de satisfazer, sendo posteriormente divididos em subcondições ou sub-objectivos; a ponderação destes objectivos e a sua soma apresentará a melhor solução, neste caso, para a produção de cartografia. Na avaliação desenvolvida pela autora privilegiaram-se as capacidades

[117] Os autores referem que o OSSIM é muito pesado – a versão completa tem 1 GB -, e, por isso, excluem-no da análise logo na primeira fase. Compreende-se esta decisão porque os testes foram executados num computador Pentium III 450 MHz, 328 Mb RAM com o Microsoft Server Windows 2000 (*Service Pack* 4) instalado. Ao contrário, com um Pentium Dual Core com um mínimo de 4 GB de memória RAM, executar um programa de 1 GB é relativamente pacífico.

gráficas dos vários *software*, tendo em vista a escolha daquela que apresenta maior potencialidade para produzir cartografia de acordo com os cânones da semiologia gráfica.

Por fim, num outro estudo que destacamos, Eldrandaly & Naguib (2013) vão mais longe ao desenvolverem um sistema (*software*) para avaliação de *software* SIG. A sua aplicação baseia-se, mais uma vez, nos fundamentos da AMC e no Processo Analítico de Hierarquização (PAH) integrados pelo *Component Object Model* (COM)[118].

O conceito da aplicação (Figura 40) subdivide-se em três fases:

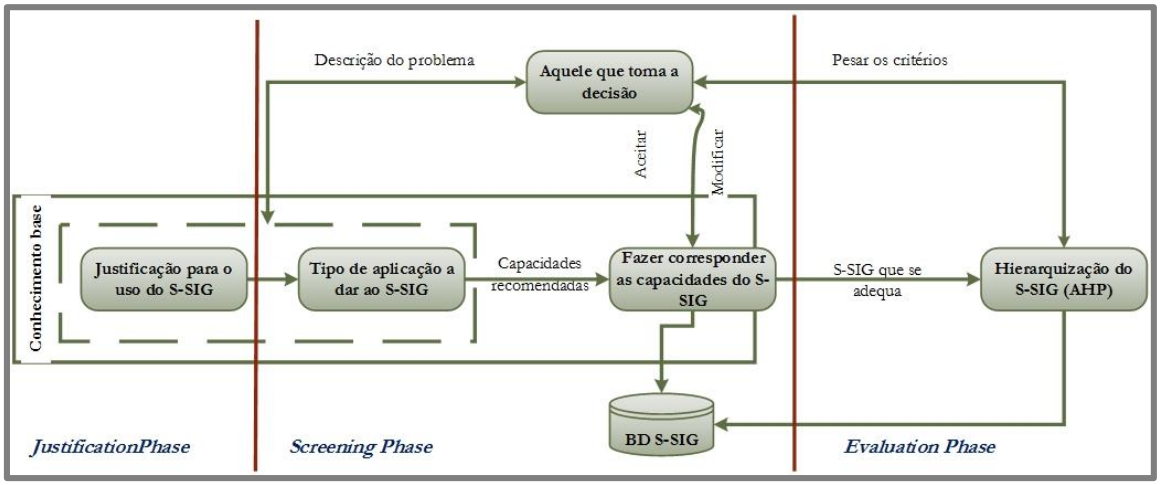


Figura 40 - Aplicação que integra os conceitos de AMC-PAH-COM, tendo em vista a selecção de *software* SIG. Fonte: Eldrandaly & Naguib (2013), p. 154.

i. *Justification Phase* – o utilizador indica a aplicação que quer dar ao *software* SIG (que problema quer ver resolvido), esperando uma resposta do sistema à pergunta 'Um *Software* SIG funcionará para o problema considerado?';

ii. *Screening Phase* – o sistema assiste o utilizador na identificação de *software* SIG que se ajusta ao projecto e ao problema (selecção a partir de propriedades genéricas do projecto e do *software*);

iii. *Evaluation Phase* – depois de identificar os pacotes com capacidades para a resolução do problema, através da especificação de critérios de interesse que não foram explicitados na fase anterior, recorrendo à AHP e AMC, a aplicação indica o *software* que cuja arquitectura

[118] A COM é uma tecnologia que garante a interoperabilidade entre componentes dos sistemas, assume-se como a capacidade de fazer as componentes de um ou vários *software* cooperar e comunicar entre si, independentemente de estarem ou não programadas em diferentes linguagens, da sua interface ou plataforma onde são executadas.

melhor se ajusta ao âmbito de determinado projecto que começa por ter de responder à pergunta da fase i).

Tal como demonstram as referências citadas, a AMC é uma técnica viável quando o objectivo é a escolha e selecção de *software*, sendo recursivamente, e até intuitivamente (não se usam os termos e conceitos formais), utilizada pelos vários autores nos seus ensaios.

Genericamente e em teoria, a AMC é uma sequência de raciocínios que estabelecem uma classificação ordenada de intensidade ou predisposição de todas as alternativas envolvidas no processo de tomada de decisão relativamente a uma meta previamente estabelecida, com base num conjunto de critérios devidamente valorados e ponderados. Por outras palavras, trata-se de uma metodologia de análise que faz com que um conjunto de alternativas seja avaliado com base num conjunto de critérios conflitivos e não comparáveis, de modo a produzir um resultado que se considera a "melhor solução" para o problema (Ferreira, 2001). Assim, a estrutura hierarquizada da AMC assenta, segundo Malczewski (1999) em seis componentes matriciais (Figura 41), a saber:

i. A análise inicia-se com a definição de uma meta que um grupo de decisores pretende materializar e concretizar;

ii. Esta meta é imposta/definida por um indivíduo, grupo ou grupos de interessados (agentes decisores), que têm objectivos e preferências relativamente à avaliação dos critérios;

iii. Cada agente decisor sugere um conjunto de objectivos e de atributos, aqueles que depois de valorados, ponderados e combinados indicam as melhores alternativas para se cumprir o objectivo. Neste esquema, o critério é considerado um termo genérico que engloba os conceitos de objectivo e atributo.

iv. As alternativas complementam a matriz de decisão (correspondem às linhas) e definem o universo de decisão ou variáveis de acção;

v. A valoração dos atributos atribuída a cada alternativa em associação com as preferências (peso que representa a maior ou menor importância desse atributo) na tomada de decisão produz um conjunto de saídas. A sua comparação permite averiguar qual ou quais as melhores alternativas para concretizar a meta;

vi. Por fim, envolvendo todo o processo encontra-se o contexto ambiental, contexto momentâneo em que a decisão está a ser processada (aqui, a noção de 'ambiental' é aplicada em sentido lato).

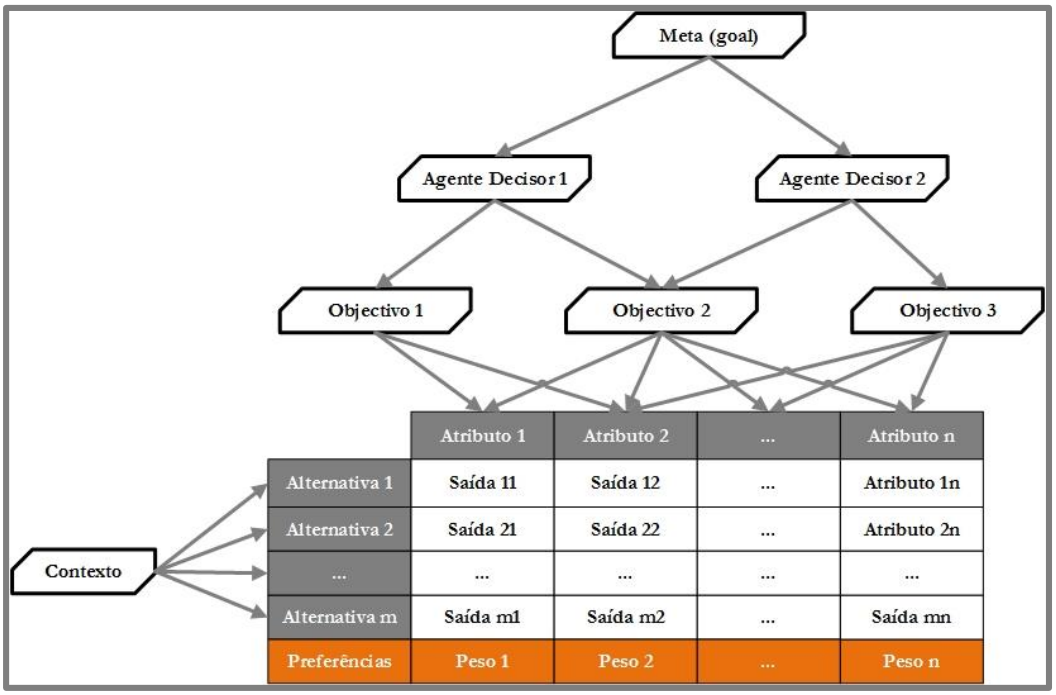


Figura 41 - Estrutura da Análise Multicritério. Adaptado de Malczewski (1999), p. 82.

Este último conceito (contexto ambiental) reflecte o grau de incerteza dos resultados da análise em cada uma das alternativas. Assim, para cada uma delas, existe um conjunto de resultados possíveis que dependem do contexto ambiental. De um modo geral, podemos afirmar que é este conceito que multiplica as possíveis estruturas de análise para um mesmo problema, pois, embora a meta seja a mesma, o contexto da tomada de decisão normalmente não o é – por isso, apesar do nosso objectivo ser idêntico ao de Chen *et al.* (2010), a estrutura da nossa AMC não o será exactamente, visto que o contexto é outro. Por outras palavras, a AMC é um processo indutivo que impele a construção de um modelo específico para cada problema em cada momento, portanto, embora os modelos possam apresentar uma estrutura processual semelhante, são operacionalmente singulares, isto é, são construídos de forma semelhante mas a sua aplicação a cada realidade implica ajustamentos específicos ao nível da definição dos objectivos, atributos e juízos (ponderações) que são utilizados ao longo da execução do modelo.

Assim, em termos operacionais e de implementação, o esquema da AMC segue necessariamente a seguinte ordem de etapas (expressas na Figura 42):

i. Regras de valoração dos atributos;

ii. Regra de decisão - ponderação dos atributos;

iii. Regra de decisão – agregação dos atributos num único valor escalar indicativo da predisposição de cada alternativa para a concretização da meta.

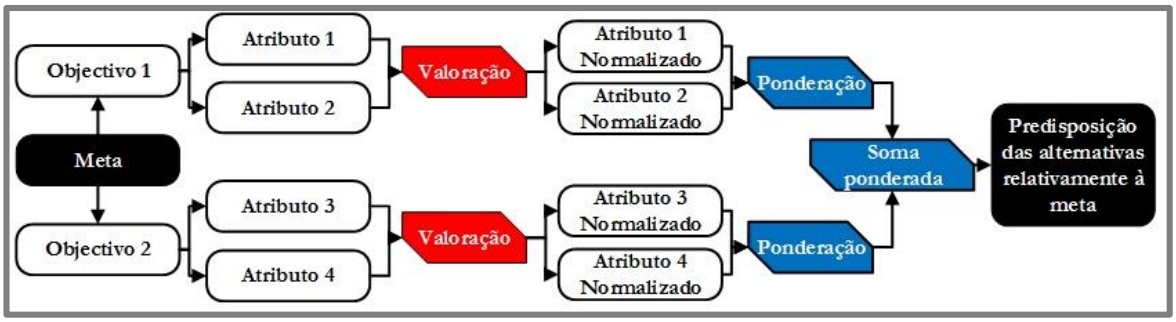


Figura 42 - Operacionalização de um processo tipo de Análise Multicritério.

A execução de qualquer AMC direccionada a uma meta implica, antes de mais, o estabelecimento de várias regras de valoração, uma vez que, e como se disse, os critérios em análise são conflituantes, dispares e têm uma expressão estatística dissemelhante. Então, de modo a obter um único valor escalar (medido numa unidade abstracta – *score*) que represente a predisposição das várias alternativas relativamente à meta, temos de comparar o que, à partida, não podia ser comparado, porque as unidades de medida ou os tipos de dados dos atributos não são os mesmos. Isto consegue-se com a normalização da expressão estatística dos atributos, tendo em vista a obtenção de um *score* que expresse a orientação desses parâmetros relativamente à condição imposta pelo objectivo.

Existem vários tipos de normalização (*standardization*) consoante o tipo de dados estatísticos disponíveis, destacam-se alguns referidos por Malczewski (1999):

i. *Ranking* para escalas qualitativas ordinais;

ii. *Rating* para dados qualitativos nominais sem ordem aparente;

iii. *Maximum Score* ou o *Range Score* para transformações lineares em escalas de dados quantitativos contínuos[119];

[119] O *Maximum Score* assegura a proporcionalidade dos dados base, mas o valor mais baixo obtido pode não ser 0, o que pode ter consequências na interpretação dos resultados. Expressa-se da seguinte maneira: maximização do critério (utiliza-se quando o maior valor do atributo significa maior ajustamento ao objectivo) - $x'_{ij} = \frac{x_{ij}}{x_j^{max}}$; minimização do critério (usa-se quando o menor valor do atributo significa maior ajustamento ao objectivo) - $x'_{ij} = 1 - \frac{x_{ij}}{x_j^{max}}$. Por sua vez, o *Range Score* assegura que os *scores* mínimo e máximo sejam sempre, respectivamente, zero e um (o que facilita interpretação), no entanto, não assegura a proporcionalidade directa com os valores base. Expressa-se da seguinte maneira: maximização do critério - $x'_{ij} = \frac{x_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}}$; minimização do critério: $x'_{ij} = \frac{x_j^{max} - x_{ij}}{x_j^{max} - x_j^{min}}$.

Com todos os atributos normalizados de acordo com a sua orientação relativamente ao objectivo/meta[120], resta-nos agregá-los, tendo em vista a obtenção de um único valor escalar final para cada alternativa, representativo do grau de predisposição da última para o cumprimento da meta. No caso de acharmos que todos os critérios têm o mesmo peso na decisão, poder-se-ia somar os vários valores normalizados, ou, então, calcular a sua média. No entanto, na maior parte das vezes, os critérios têm de ser pesados, pois uns são mais importantes que outros na tomada de decisão. A soma ponderada é a regra de decisão[121] mais comum e mais utilizada, permitindo agregar os valores normalizados dos vários atributos num *score* final, não deixando de respeitar o peso de cada um.

2.2. A relação entre Análise Multicritério e o Processo Analítico de Hierarquização: valoração e ponderação dos critérios

Como se disse, neste exame são envolvidos critérios que rivalizam entre si, pois apontam para soluções distintas. O importante é trabalhá-los com o intuito de obter um compromisso razoável entre as soluções obtidas. Nesse sentido, os vários critérios têm de passar por um processo de priorização, assunto complexo que, inevitavelmente, envolve a inclusão de um certo grau de subjectividade no processo de decisão (Ferreira, 2001).

O PAH, desenvolvido por Saaty (1988), é um método de comparação por pares que, apesar das suas limitações, evidencia a grande vantagem de possibilitar um certo controlo sobre o grau de (in)coerência das ponderações derivadas.

Para além disto, tem também o grande benefício de poder ser usado tanto para a valoração dos atributos como para a sua ponderação, juntando-se às outras regras de valoração acima referidas. No caso específico deste trabalho, uma técnica como a PAH ajusta-se perfeitamente às nossas necessidades, visto que a valoração a designar a cada atributo depende da relação entre a alternativa e seus pares. Por exemplo, se tivermos como atributo o nível de complexidade de um *software*, faz mais sentido analisar-se a complexidade da alternativa A, tendo em consideração a complexidade das restantes. A juntar a isto, normalmente, os atributos usados na comparação de *software*, tal como se vê nos trabalhos referidos, não têm expressão quantitativa, nem sequer ordinal – dependem

[120] Por regra, os atributos são valorados de maneira a que os valores mais elevados resultantes dessa operação signifiquem que essa alternativa garante o cumprimento do objectivo/meta.
[121] A regra de decisão integra-se no conceito mais amplo de estratégia de decisão (que inclui todos os procedimentos necessários para se atingir uma determinada solução), e constitui um instrumento matemático para comparar e seleccionar as alternativas com base num processo prévio, de avaliação (Ferreira, 2001).

sobretudo da capacidade do operador que as analisa e compara segundo os conhecimentos de que dispõe sobre elas e sobre as especificações que tem de respeitar.

Por outro lado, os pesos de cada atributo podem também ser conseguidos através da contraposição dos vários pares, discriminando que A é mais importante que B, mas também que este último é mais importante que C.

Em termos práticos, o PAH parte de uma matriz composta por linhas e colunas relativas aos vários critérios a ponderar, a partir da qual se estabelece uma análise comparativa entre a importância de cada par de critérios (Tabela XXI).

Tabela XXI - Estrutura da matriz de relação por pares. Adaptado de Saaty (1988), p. 26 e Ferreira (2001), p. 98

Critérios	C_1	C_2	C_3	…	C_n
C_1	1	w_{21}	w_{31}	…	w_{n1}
C_2	w_{12}	1	w_{32}	…	w_{n2}
C_3	w_{13}	w_{23}	1	…	w_{n3}
…	…	…	…	…	…
C_n	w_{1n}	w_{2n}	w_{3n}	…	1

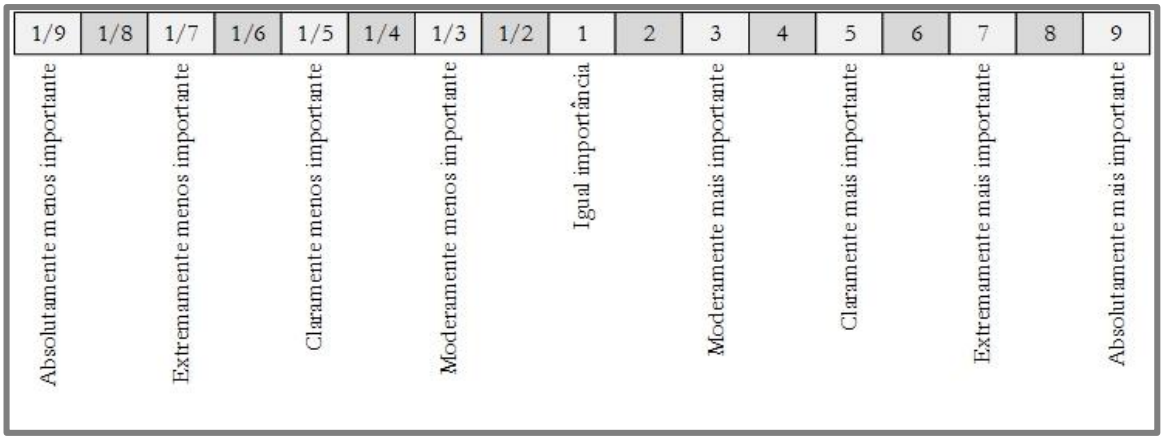


Figura 43 - Escala de comparação para os pares de critérios. Fonte: Ferreira (2001), p. 98, adaptado de Saaty (1988).

De acordo com Ferreira (2001), interpretando Saaty (1988), a atribuição das prioridades entre os critérios é feita a partir de um juízo valorativo com base numa escala que varia de 1 a 9 (ou 1 a $1/9$), na qual o valor 1 indica que ambos os critérios ou alternativas apresentam igual importância,

em contraposição, o valor 9 (ou $^1/_9$) assinala uma situação em que um dos factores é absolutamente mais (ou menos) importante que o outro (Figura 43)[122].

A compreensão do descrito sobre o PAH é suficiente para o incrementar. A página *Business Performance Management* (http://bpmsg.com/, acedido a 1 Novembro, 2015) disponibiliza gratuitamente uma macro do Microsoft Excel em formato *.xlsx* que permite ao utilizador comparar pares de critérios e/ou alternativas. Começa-se por definir o número de critérios que queremos considerar (a aplicação restringe-nos a 10 critérios), o número de participantes (até 20) e explicitam-se os critérios (Figura 44).

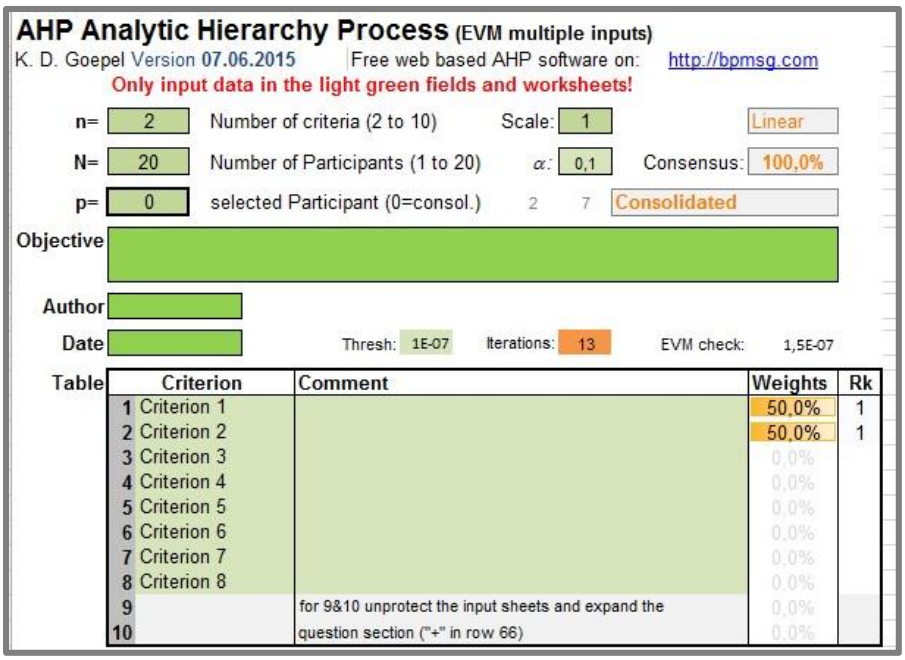
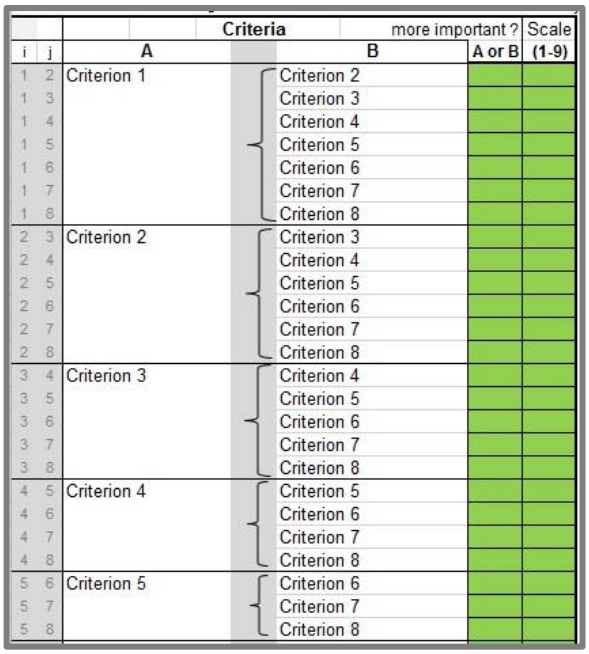


Figura 44 - Página inicial da macro do Microsoft Excel desenvolvida para executar o Processo Analítico de Hierarquização. Fonte: http://bpmsg.com/new-ahp-excel-template-with-multiple-inputs/ (acedido em 1 Novembro, 2015).

De seguida, numa tabela como a apresentada na Figura 45, o utilizador avalia cada par possível de critérios, discriminando qual é que acha mais importante e quão mais importante é relativamente ao outro. Neste ponto, a macro está preparada para formalizar aritmeticamente os juízos expressos pelo operador segundo a determinação do vector-próprio principal normalizado da matriz de relação (Saaty, 1988). Como as relações contidas na matriz assumem um carácter recíproco, isto é,

[122] Segundo Ferreira (2001), na obra de Saaty (1988) o autor explicita apenas os valores da escala apresentados na figura acima, considerando os valores 2, 4, 6 e 8 (assim como $^1/_2$, $^1/_4$, $^1/_6$, $^1/_8$) como níveis intermédios utilizáveis em situações em que é necessário chegar a um compromisso de valoração.

se o critério C_1 é três vezes mais importante do que o critério C_2, então este último é $^1/_3$ menos importante que o primeiro, a determinação do vector-próprio principal da matriz pode ser feita através da divisão dos elementos da coluna pelo somatório da coluna respectiva e, posteriormente, somando os valores em linha e dividindo-os pelo número total de critérios (Ferreira, 2001) – Tabela XXII.



		Criteria		more important ?	Scale
i	j	A	B	A or B	(1-9)
1	2	Criterion 1	Criterion 2		
1	3		Criterion 3		
1	4		Criterion 4		
1	5		Criterion 5		
1	6		Criterion 6		
1	7		Criterion 7		
1	8		Criterion 8		
2	3	Criterion 2	Criterion 3		
2	4		Criterion 4		
2	5		Criterion 5		
2	6		Criterion 6		
2	7		Criterion 7		
2	8		Criterion 8		
3	4	Criterion 3	Criterion 4		
3	5		Criterion 5		
3	6		Criterion 6		
3	7		Criterion 7		
3	8		Criterion 8		
4	5	Criterion 4	Criterion 5		
4	6		Criterion 6		
4	7		Criterion 7		
4	8		Criterion 8		
5	6	Criterion 5	Criterion 6		
5	7		Criterion 7		
5	8		Criterion 8		

Figura 45 - Captura de ecrã de um Tabela que permite ao utilizador atribuir juízos relativos a cada par possível de critérios, dizendo, em cada par, qual é o mais importante e quão mais importante é. Fonte: macro do Microsoft Excel desenvolvida para executar o Processo Analítico de Hierarquização - http://bpmsg.com/new-ahp-excel-template-with-multiple-inputs/ (acedido em 1 Novembro, 2015).

Tabela XXII - Determinação das ponderações pelo método do vector-próprio. Fonte: Ferreira (2001), p. 99

Critérios	C_1	C_2	C_3	…	C_n	W_i
C_1	$1/\sum C_1$	$w_{21}/\sum C_2$	$w_{31}/\sum C_3$	…	$w_{n1}/\sum C_n$	$\sum C_1/n$
C_2	$w_{12}/\sum C_1$	$1/\sum C_2$	$w_{32}/\sum C_3$	…	$w_{n2}/\sum C_n$	$\sum C_2/n$
C_3	$w_{13}/\sum C_1$	$w_{23}/\sum C_2$	$1/\sum C_3$	…	$w_{n3}/\sum C_n$	$\sum C_3/n$
…	…	…	…	…	…	…
C_n	$w_{1n}/\sum C_1$	$w_{2n}/\sum C_2$	$w_{3n}/\sum C_3$	…	$1/\sum C_n$	$\sum C_n/n$
	$\sum C_1$	$\sum C_2$	$\sum C_3$	…	$\sum C_n$	

Os resultados desta operação foram apontados nos campos representados na Figura 44, encontrando-se acompanhada por um conjunto de indicadores que expressam o grau de consistência dos pesos definidos, falamos essencialmente da Razão de Consistência (CR na Figura 46). Segundo Ferreira (2001), a Razão de Consistência é calculada com base no quociente entre o Índice de consistência ($IC = \frac{(\lambda_{max} - n)}{n-1}$) e o Índice Aleatório[123], e deverá apresentar um valor inferior a 0,10 (10%) para que os juízos feitos sobre os critérios na matriz de avaliação possam ser considerados consistentes.

Figura 46 - Apresentação da Razão de Consistência, indicador que testa o grau de consistência do Processo Analítico de Hierarquização. Fonte: macro do Microsoft Excel desenvolvida para executar o Processo Analítico de Hierarquização - http://bpmsg.com/new-ahp-excel-template-with-multiple-inputs/ (acedido em 1 Novembro, 2015).

Em suma, o PAH, aplicado durante a operacionalização de uma AMC, é uma ferramenta de grande utilidade na ponderação dos vários critérios, podendo ser adaptada para comparar as diferentes alternativas (que substituem os critérios), ou seja, os vários *software*. Esta possibilidade e o facto de haver indicadores que nos garantem a consistência do processo foram factores preponderantes na adopção desta técnica para seleccionarmos o *software* a testar em cada exercício.

[123] O Índice Aleatório corresponde a um índice de consistência calculado para uma matriz gerada aleatoriamente, com ordens que variam entre 1 e 15 (Ferreira, 2001). Saaty (1988, p. 21) apresenta os valores obtidos:

N=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IA=	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49	1,51	1,48	1,56	1,57	1,59

Capítulo II – Exercício I: Produção Automática de Cartografia de Susceptibilidade à ocorrência de Movimentos de Vertente

Entendidos na perspectiva mais divulgada, designadamente, por autores anglo-saxónicos (Varnes, 1978, 1984; Hutchinson, 1988; Cruden & Varnes, 1996; Santos 2002 e 2013, entre outros) e aceites pela própria *Working Party of World Landslide Inventory* os movimentos de vertente são fenómenos que apresentam uma dinâmica comandada exclusivamente pela gravidade – newtonianos, portanto, que, no seio dos Riscos Naturais representam uma das grandes ameaças para as populações. A vulnerabilidade crescente dos territórios face a este tipo de fenómenos tem vindo a traduzir-se pela produção de consequências, por vezes catastróficas que implicando (ou não) a perda de vidas humanas, quase sempre geram danos materiais bastante avultados. Neste sentido, e porque o seu estudo se enquadra no âmbito das ciências cindínicas, o enfoque deve estar a montante, na identificação e inventariação dos movimentos, e na definição de estratégias de *Early Warming* que permitam antecipar a sua ocorrência – no espaço e no tempo, permitindo a mitigação dos seus impactes nas sociedades potencialmente ameaçadas (Patriarca *et al.*, 2014b).

Em Portugal, directa ou indirectamente, os movimentos de vertente são contemplados em diversos diplomas, tais como as Leis de Base da Protecção Civil (Lei n.º 27/2006) encarnadas na ANPC, ou no regime jurídico da Reserva Ecológica Nacional (Decreto-Lei n.º 166/2008). A nível local, o Plano Municipal de Emergência (PME), elaborado pela Comissão Municipal de Protecção Civil e contemplado na Lei n.º 65/2007, prevê a construção obrigatória de uma carta de risco e um plano prévio de intervenção para cada tipo de risco existente no município, entre os quais pode estar o risco de ocorrência de movimentos de vertente.

É neste contexto de elaboração ou revisão do PME que o primeiro exercício de *benchmarking black-box* é realizado, tentando esclarecer as instituições que se preocupam com estas questões sobre a apetência e limitações de várias soluções SIG previamente seleccionadas através da implementação de uma AMC ajustada à especificidade deste ensaio, mas entre as quais consta uma proprietária (ArcGIS 10.2).

Posto isto, apresentam-se os tópicos que sistematizam o que vamos desenvolver nesta parte do trabalho:

1. **Técnicas de análise da susceptibilidade à ocorrência de movimentos de vertente - método do valor informativo: bases terminológicas e sequência processual em ambiente SIG**
   i. Esclarecer, de modo sucinto e breve, conceitos-base das ciências cindínicas.
   ii. Enquadrar o método do valor informativo relativamente a outras metodologias de avaliação da susceptibilidade de ocorrência de movimentos de vertente.
   iii. Delimitar a área de estudo.
   iv. Estabelecer uma linha processual e operacional que identifique as várias etapas do exercício de produção de cartografia de susceptibilidade.
   v. Justificar e explicar cada etapa processual.
2. **Base analítica para selecção de *software* com funcionalidades para implementação do método do valor informativo, construção e caracterização dos procedimentos destinados à realização das etapas estabelecidas**
   i. Organizar um conjunto de critérios que respondam à necessidade de selecção de alternativas SIG a testar.
   ii. Valorar e ponderar os vários atributos e objectivos.
   iii. Apresentar os *software* seleccionados.
   iv. Caracterizar os Programas/rotinas construídas com o fim de produzir cartografia de susceptibilidade nos vários *software* previamente seleccionados.
3. **Comparação de desempenho entre *Software* SIG Proprietário e *Software* SIG Livre: 1ª Análise**
   i. Apresentar e discutir os resultados de performance obtidos na execução dos vários Programas.
   ii. Apontar possíveis limitações, erros ou falhas.
   iii. Relacionar os resultados com a estrutura lógica e características dos vários procedimentos/rotinas.

I. Técnicas de análise da susceptibilidade à ocorrência de movimentos de vertente – método do valor informativo: bases terminológicas, lógica e sequência processual em ambiente SIG

> *"Desastres, especialmente aqueles conotados pela opinião pública como catástrofes naturais, não são a maior ameaça à humanidade. Apesar da conotação destrutiva que se atribui a terramotos, epidemias e fomes, uma grande parte da população mundial tem a sua vida encurtada por eventos não noticiados"*
>
> P. Blaikie; T. Cannon; I. Davis & B.Wisner (1994:3)[124]

Entendendo o conceito de "perigo" como o processo ou evento com potencial para produzir perdas e danos identificados, e o de "risco" como a probabilidade de ocorrência de um processo perigoso e respectiva estimativa das suas consequências sobre pessoas, bens ou ambiente (os elementos "em risco"), expressas em danos corporais e/ou prejuízos materiais e funcionais directos ou indirectos, o conceito de "susceptibilidade" diz respeito à incidência espacial do perigo, ou seja, está a montante do risco, e representa a propensão de uma área para ser afectada por um determinado perigo, em tempo indeterminado, sendo avaliada através dos factores de predisposição para a ocorrência dos processos ou acções, não contemplando o seu período de retorno ou probabilidade de ocorrência – hierarquização do território quanto à predisposição para ser afectado pelo perigo (Smith, 2004; Julião *et al*., 2009).

Esclarecida a perspectiva que vamos assumir na apresentação destes conceitos relacionados com as ciências cindínicas e averiguada que está a pertinência deste exercício para organismos específicos da APP, devemos dizer que este ensaio aplica uma das várias alternativas metodológicas no âmbito da produção de cartografia de susceptibilidade relativa à ocorrência de movimentos de vertente, tendo em vista, como se disse, a comparação entre vários *software* SIG.

---

[124] No original, *"Disasters, especially those that area connected in the minds of the public with natural hazards, are not the greatest threat to humanity. Despite the lethal reputation of earthquakes, epidemics, and famines, many more of the world's population have their lives shortened by unnoticed events…". In* BLAIKIE, P.; CANNON, T.; DAVIS, I. & WISNER, B. (1994) – **At risk: natural hazards, people's vulnerability and disasters**. Routledge, 1ª Edição, Nova Iorque, 284 p.

Para o efeito, definiu-se como área de trabalho o espaço delimitado pela área abrangida pela folha n.º 126 da Carta Militar de Portugal do ex-Instituto Geográfico do Exército, com escala 1:25 000 (Figura 47). A escolha recaiu sobre este território específico por já o termos estudado e trabalhado em ensaios anteriores (Santos, 2002 e 2013; Patriarca *et al*., 2014b), tratando-se de uma área considerada de elevada susceptibilidade à ocorrência de movimentos de vertente.

Como se sabe, qualquer exercício de modelação com propósitos prospectivos toma como variável dependente um inventário de registo de ocorrência do fenómeno que, neste caso, diz respeito ao inventário de movimentos de vertente efectuado em 2001 e publicado no estudo de Santos (2002). Desta BD constam 441 ocorrências, que o autor classificou em três tipos (Grau 1, Grau 2 e Grau 3) em conformidade com os parâmetros morfodinâmicos propostos pela *Working Party of World Landslide Inventory*.

Para medir e prever a maior ou menor predisposição do território para a ocorrência deste fenómeno explorou-se uma metodologia que consiste na aplicação em ambiente SIG (por via de programação) do Método do Valor Informativo (VI), considerando-se os seguintes factores condicionantes (variáveis independentes) – que explicam o fenómeno:

i. Declives;

ii. Inverso do índice topográfico;

iii. Perfil transversal (curvatura da vertente);

iv. Características litológicas;

v. Ângulo de exposição das vertentes;

vi. Ocupação e uso do solo.

Uma vez definidos os parâmetros de entrada do modelo de avaliação da susceptibilidade, procuramos estabelecer um conjunto de etapas sequenciais, pois, como se sabe, qualquer processo de modelação obriga à execução de vários tipos de procedimentos que visam a preparação adequada dos dados de entrada, a implementação das técnicas que têm como fim a obtenção do tipo de cartograma pretendido, e a avaliação da exactidão do resultado final. Neste sentido, a Figura 48 discrimina as várias etapas ordenadas.

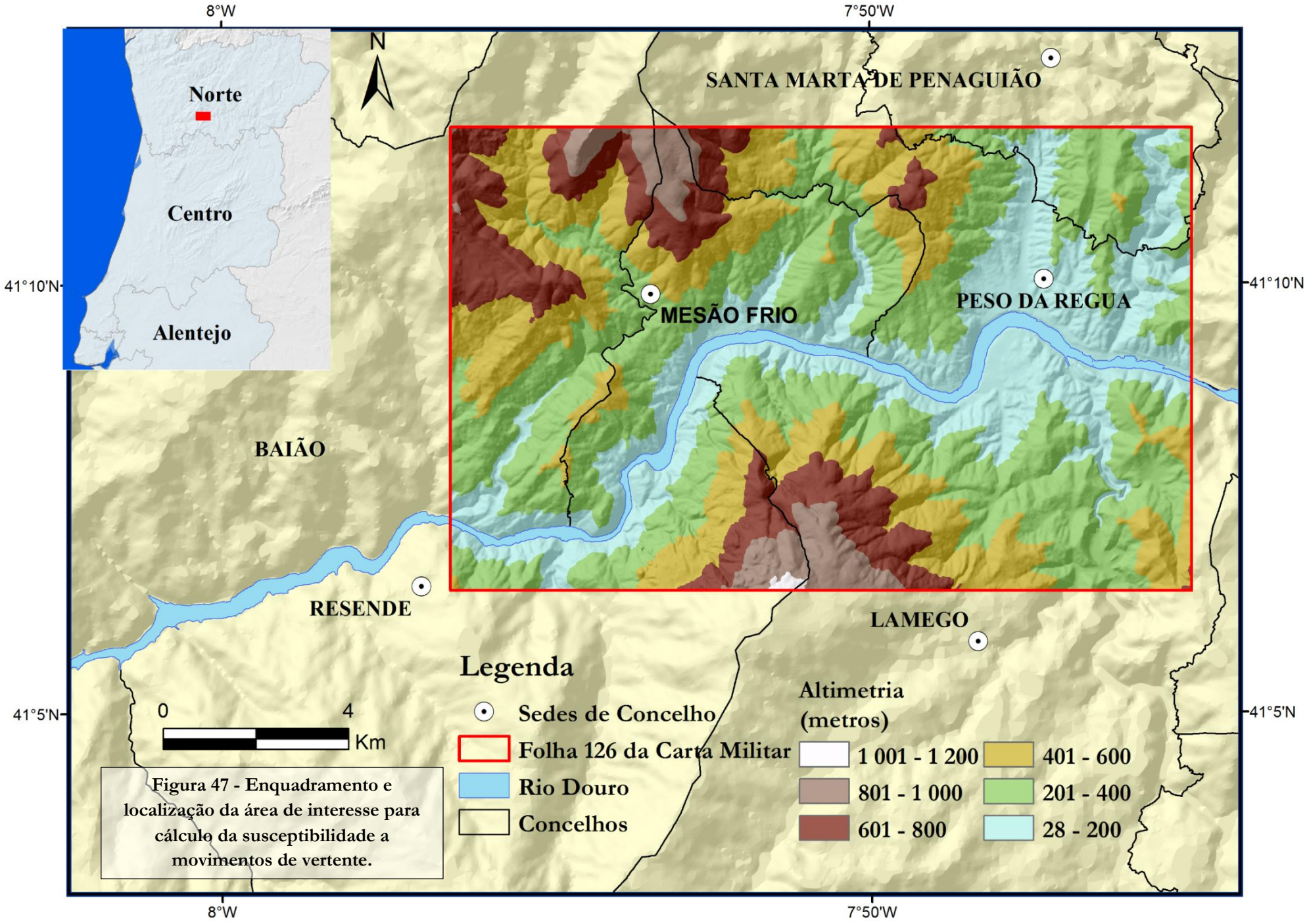


Figura 47 - Enquadramento e localização da área de interesse para cálculo da susceptibilidade a movimentos de vertente.

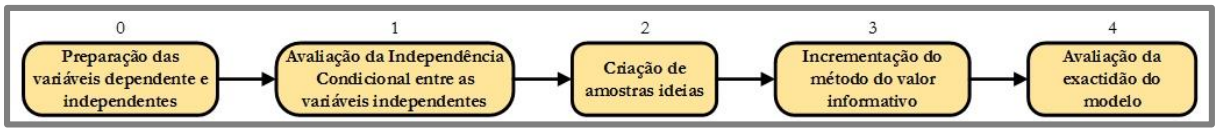


Figura 48 - Sequência de etapas envolvidas na implementação do método do valor informativo, tendo em vista a produção de cartografia de susceptibilidade a movimentos de vertente.

Destaca-se que a fase de preparação dos dados de entrada não foi alvo de avaliação comparada entre *software*, pois julgamos que para o teste de eficiência (e respectiva comparação dos *outputs*) ser válido deve-se partir exactamente dos mesmos dados de entrada. Portanto, recorreu-se a um único *software* (ArcGIS 10.2) para gerar estes elementos de partida.

1.1. Preparação da variável dependente e das variáveis independentes

Em primeiro lugar, e no que diz respeito à variável dependente, por não se tratar de um estudo de geomorfologia, fomos menos exigentes e ignoramos a existência de várias categorias de movimentos (G1, G2 e G3), tendo estas sido agrupadas num único tema pontual. *A priori*, como o método do VI pressupõe a discriminação do somatório da extensão de área onde se registaram ocorrências - somatório que, por um lado, se refere à extensão de área onde se registaram ocorrências que se intersecta com cada classe de cada factor, e, por outro, se refere à total extensão de área onde se registou ocorrência do fenómeno -, gerou-se uma circunferência (*buffer*) sobre cada *feature* do tema vectorial que representa a existência passada do fenómeno (Programa 15[125]).

Por sua vez, a preparação dos factores independentes está inevitavelmente condicionada pela construção prévia do Modelo Digital de Terreno (MDT), elemento fulcral em todo o processo de modelação, visto que um número expressivo de variáveis são dele derivadas. Estamos conscientes de que a qualidade do MDT depende, em primeiro lugar, da qualidade dos dados de entrada, e, em segundo lugar, do método utilizado. A existência de erros está, antes de tudo, dependente da escolha do método de construção do MDT, como bem nos transmitem autores como Weibel & Heller (1991), Wise (1998), El-Sheimy *et al.* (2005) ou Matos (2008).

Estes mesmos autores referem dois modos essenciais para produção de um MDT: interpolação (estimativa do valor da superfície nos locais onde não existem pontos da amostra) e construção de uma Rede Irregular de Triângulos (RIT – TIN do inglês). Como ambos envolvem pressupostos matemáticos que não nos atrevemos a discutir, construímos dois modelos em formato de dados

[125] **Ver Anexo A, p. A13.**

matricial que esboçam o relevo da nossa área de estudo, um é o resultado da execução da ferramenta *Arctoolbox* > *Spatial Analyst Tools* > *Interpolation* > *TopoToRaster* (algoritmo de interpolação desenhado especificamente para a criação de MDT válidos hidrologicamente, baseado no programa ANUDEM desenvolvido por Michael Hutchinson)[126] e o outro é o *output* da ferramenta *Arctoolbox* > *3D Analyst Tools* > *Data Management* > *TIN* > *Create TIN* depois de convertido para formato matricial (Programa 16[127]). A subtracção das duas matrizes revelou que existem diferenças até 30 metros, no entanto, embora 88,28% dos pixéis tenham valores distintos, cerca de 96% das diferenças são inferiores a 5 metros. Isto vai ao encontro daquilo que está documentado sobre as desigualdades destes dois métodos (Wise 1998), e em circunstâncias normais justificaria a avaliação da exactidão temática das duas imagens geradas; contudo, apercebemo-nos que as exposições das vertentes derivadas a partir do MDT interpolado (*TopoToRaster*) eram incongruentes com a realidade, salientando-se áreas não planas ao longo do curso do Rio Douro. Consequentemente, assumimos que o método de Triangulação de Delaunay é o mais ajustado aos dados que tínhamos disponíveis.

A partir do MDT, derivaram-se quatro dos factores independentes (declives, exposição de vertentes, inverso do índice topográfico[128] e perfil transversal (curvatura)[129] segundo os processos indicados no Programa 17[130], que tem por base as indicações de Garcia (2012) no que diz respeito ao cálculo do inverso do índice topográfico e perfil transversal.

---

126 Informação que consta do manual do utilizador do *software* ArcGIS 10.2 - http://resources.arcgis.com/EN/HELP/MAIN/10.2/index.html#//009z0000007m000000 (acedido em 2 Dezembro, 2015).

127 **Ver Anexo A, p. A13.**

128 O conteúdo em água ou grau de saturação dos terrenos é reconhecidamente um parâmetro importante na ocorrência de movimentos de vertente (Santos, 2002; Garcia, 2012). Este parâmetro pode ser obtido indirectamente através do índice topográfico, assumindo que este serve como indicador da quantidade de água no terreno. Este índice relaciona a área de contribuição, a montante de um ponto, com os perfis de vertente associados, expresso pela seguinte fórmula: $WI = In\ (A_s / \tan \beta)$, sendo A_s a área de contribuição específica (resulta da relação entre a área de contribuição e a largura da célula na direcção da vertente) e β o declive do local (Garcia, 2012). No entanto, a utilização da fórmula original levanta problemas no caso da presença de áreas planas. Assim, para evitar os erros derivados da divisão por 0, utilizou-se o inverso do índice topográfico, obtido pela expressão $IWI = (\beta / A_s)$ (*idem, ibidem*).

129 A inclusão deste elemento no modelo de averiguação da susceptibilidade justifica-se por expressar o nível de concentração de água nas vertentes. De acordo com Garcia (2012), o perfil das vertentes pode ser analisado de três modos distintos: i) perfil longitudinal (*profile curvature*), que expressa a forma da vertente na direcção da sua maior inclinação indicando o modo como o declive se comporta de montante para jusante na vertente; ii) o perfil transversal (*plan curvature*), que indica a forma da vertente numa orientação perpendicular à do perfil longitudinal, expressando as alterações da exposição da vertente; e iii) a curvatura (*tangencial curvature*), cujo resultado é a combinação das duas anteriores. A opção lógica seria utilizar a curvatura, porém, segundo este autor, o grau de ajuste destas componentes relativamente ao modelo pode ser distinto, pelo que se deve usar aquele que melhor se ajusta à especificidade do nosso problema. Neste exercício, como o nosso objectivo é a comparação entre *software* e não propriamente a análise de susceptibilidade, optámos por utilizar a curvatura por ser a combinação dos outros dois.

130 **Ver Anexo A, p. A14.**

A informação relativa ao uso e ocupação do solo foi retirada do *Corine Land Cover* de 2006 disponibilizado pela DGT[131], sendo posteriormente agrupada nas seguintes classes:

i. Espaços agrícolas;
ii. Folhosas;
iii. Vinhas;
iv. Espaços urbanos;
v. Florestas abertas;
vi. Herbáceas;
vii. Florestas mistas;
viii. Florestas de resinosas;
ix. Pastagens;
x. Pomares;
xi. Corpos de água.

Por fim, as classes litológicas foram extraídas a partir da vectorização sobre a folha n.º 10-C da Carta Geológica de Portugal, com escala de 1:50 000.

Todos estes temas foram convertidos para o formato matricial, no qual o espaço é segmentado em células uniformes contíguas com um mesmo valor de área (resolução, do inglês *cellsize*), e em que cada uma destas unidades assume um valor numérico, neste caso representativo de um intervalo ou propriedade de cada um dos factores em consideração. O emprego deste modelo de dados agiliza os cálculos do VI relativo a cada classe (de cada factor) e a sua posterior agregação (álgebra de mapas) num único cartograma que representa a susceptibilidade a movimentos de vertente em cada unidade territorial (célula).

De modo a garantir a interoperabilidade, escolheram-se formatos de ficheiro *open standard* - *geotif* para modelo matricial e *shapefile* para modelo vectorial.

1.2. Avaliação da Independência Condicional entre as variáveis independentes

[131] Embora este elemento cartográfico possua uma unidade mínima cartográfica de 25 hectares, a opção sobre a sua utilização resulta da sua desagregação até ao nível 3.

Com os dados de partida preparados, encontramo-nos em condições de iniciar o sufrágio que elegerá as alternativas mais performantes na execução das rotinas que serão propostas mais adiante.

Segundo Bonham-Carter (1994) *apud* Garcia (2012), análises probabilísticas assumem como pressuposto de partida a independência condicional entre os factores condicionantes, evitando que os resultados finais possam ser sobrestimados se houver dependência. Havendo dependência condicional entre dois factores, assume-se que ambos têm o mesmo poder explicativo sobre variável dependente, o que pode implicar o enviesamento dos resultados da modelação.

No presente exercício aplicam-se dois métodos para avaliar este indicador: o Rácio de Independência Condicional (RIC) e o "Novo Teste de Independência Condicional" (TAC), tendo como referência os trabalhos de Garcia (2012) e Pereira *et al.* (2012). O primeiro indica a probabilidade dos temas apresentarem independência condicional, em que um valor inferior a 1 indica uma possível dependência condicional – serão de rejeitar sobretudo os valores inferiores a 0,85; o segundo aponta a probabilidade de os temas não serem condicionalmente independentes, pelo que a leitura dos dois índices é inversa – qualquer valor maior que 50% indica a existência de alguma dependência, mas devem ser rejeitadas as hipóteses com valor obtido superior a 95% (Garcia, 2012).

A extensão *Spatial Data Modeller* (SDM) - http://www.ige.unicamp.br/sdm/ (acedido em 19 Setembro, 2015) – inclui um *script* que tem como finalidade o cálculo destes dois indicadores. Tratando-se de uma *toolbox* de código aberto, foi possível estudar o código original, derivando dele um novo algoritmo em tudo semelhante ao original, mas adaptável à sintaxe programável de qualquer plataforma SIG, num teste que tem como objectivo a avaliação da sua performance em termos da sua capacidade de leitura de imagens *raster* e acesso às suas propriedades e características.

Este algoritmo, tal como se verá aquando a apresentação dos programas que o materializam, prevê a comparação iterativa de todas as imagens e inventário de movimentos, gerando uma tabela – Tabela XXIII – que apresenta os valores do RIC, do TAC e de uma medida de independência global que resulta da combinação das duas primeiras. A Tabela XXIII é o resultado que se espera que qualquer programa que implemente este algoritmo devolva, se tiver como parâmetros de entrada os factores de predisposição e inventário de ocorrências descritos anteriormente.

Tabela XXIII - Indicadores de (In)Dependência Condicional entre os factores condicionantes na área de estudo, produto que se espera dos programas que implementem o algoritmo estipulado para o seu cálculo

Medida Global de Independência Condicional

	COS	Curvatura	Declives	Exposição	Inv. Topo.	Litologia
COS	**3,464**	26,328	6,616	18,903	55,550	58,769
Curvatura	1,511	**19,821**	3,458	13,087	49,772	53,288
Declives	2,856	24,629	**5,688**	17,342	54,119	57,416
Exposição	0,003	2,739	0,029	**0,960**	24,503	28,501
Inv. Topo.	0,020	4,905	0,123	2,084	**29,962**	34,012
Litologia	0,000	0,674	0,001	0,147	15,331	**18,910**
Rácio de Independência Condicional						
	COS	Curvatura	Declives	Exposição	Inv. Topo.	Litologia
COS	**0,009**	0,009	0,009	0,009	0,009	0,009
Curvatura	0,008	**0,008**	0,008	0,008	0,008	0,008
Declives	0,009	0,009	**0,009**	0,009	0,009	0,009
Exposição	0,005	0,005	0,005	**0,005**	0,005	0,005
Inv. Topo.	0,005	0,005	0,005	0,005	**0,005**	0,005
Litologia	0,004	0,004	0,004	0,004	0,004	**0,004**
"Novo Teste de Independência Condicional"						
	COS	Curvatura	Declives	Exposição	Inv. Topo.	Litologia
COS	**98,268**	86,836	96,692	90,548	72,225	70,616
Curvatura	99,245	**90,089**	98,271	93,457	75,114	73,356
Declives	98,572	87,686	**97,156**	91,329	72,941	71,292
Exposição	99,998	98,630	99,985	**99,520**	87,748	85,749
Inv. Topo.	99,990	97,548	99,939	98,958	**85,019**	82,994
Litologia	100	99,663	100	99,927	92,334	**90,545**

1.3. Criação de amostras ideais

Qualquer processo de modelação de índole preditiva deve ser baseado em dois conjuntos de dados independentes, o que implica, frequentemente, a divisão dos dados de ocorrência em dois subconjuntos, um para treino (70%), outro para teste/validação (30%) (Garcia, 2012; Rodrigues, 2013). A separação das observações reais não pode, no entanto, ser arbitrária, pelo que é aconselhável incrementar uma estratégia que garanta que as classes dos factores condicionantes (aquelas onde se observaram ocorrências) têm uma justa representação nos dois conjuntos independentes, preferencialmente próxima dos 70%-30%. Com esta abordagem estaremos a eliminar potenciais incongruências na interpretação da medida de exactidão global do modelo.[132]

Assim sendo, também com o fim de comparar o desempenho dos *software* SIG seleccionados, criou-se um algoritmo que visa a definição de amostras ideais para cada factor de predisposição, isto é, para cada um dos factores, em cada classe/variável, os registos serão repartidos, de modo a que 70% desse inventário passe a fazer parte da amostra de treino e o restante seja associado ao grupo

[132] Imagine-se que uma determinada classe regista 10 ocorrências do fenómeno e que esse total é usado para treino e ajuste do modelo, fazendo com que na amostra de validação essa classe não registe nenhum evento. No processo de cálculo da medida de avaliação global da exactidão global do modelo, os pixéis dessa classe serão todos considerados como falsos positivos, acrescentando erro onde não existe.

de validação. Como facilmente se depreende, as amostras geradas para o factor X ajustam-se a ele, podendo, ou não, ajustar-se aos restantes (repartição 70%-30% em todas as classes), portanto, o algoritmo devolve um relatório que descreve, para todas as amostras, o número de pontos (ocorrências) associados a determinada classe de cada factor e a cada uma das amostras (treino ou validação). A Tabela XXIV constitui um exemplo de uma das tabelas geradas para uma das amostras (havendo seis factores condicionantes, o algoritmo escreve seis quadros como este).

Tabela XXIV - Matriz de avaliação do ajuste de duas 'amostras ideais' a cada factor condicional. Resultado de uma iteração do algoritmo

Clas.	N.º Pontos – Amostra 70%	N.º Pontos – Amostra 30%	Total	% Pontos na Amostra 70%	% Pontos na Amostra 30%	*Status*
Uso e Ocupação do Solo						
0	19	9	28	67,86 %	32,14%	Aprovado
1	0	0	0			
2	247	107	354	69,77 %	30,23 %	
3	5	3	8	62,50 %	37,50 %	
4	26	12	38	68,42 %	31,58 %	
5	3	2	5	60,00 %	40,00 %	
6	0	0	0			
7	5	3	8	62,50 %	37,50 %	
8	0	0	0			
9	0	0	0			
10	0	0	0			
11	0	0	0			
Perfil de Curvatura da vertente						
1	126	54	180	70 %	30%	Reprovado
2	3	2	5	60 %	40 %	
3	6	0	6	100 %	0 %	
4	1	1	2	50 %	50 %	
5	169	79	248	68,15 %	31,85 %	
Declives						
1	6	3	9	66,67 %	33,33 %	Aprovado
2	40	12	52	76,92 %	23,08 %	
3	104	43	147	70,7 %	29,25 %	
4	109	54	163	66,87 %	33,13 %	
5	41	22	63	65,08 %	34,92 %	
6	4	1	5	80 %	20 %	
7	0	0	0			
8	0	0	0			
9	0	0	0			
Exposição das vertentes						
1	1	0	1	100 %	0 %	Aprovado
2	38	13	51	74,51 %	25,49 %	
3	15	22	37	40,54 %	59,46 %	
4	58	22	80	72,50 %	27,50 %	
5	63	23	86	73,26 %	26,74 %	
6	47	13	60	78,33 %	21,67 %	
7	25	16	41	60,98 %	39,02 %	

8	36	18	54	66,67 %	33,33 %	
9	21	8	29	72,41 %	27,59 %	
Inverso do Índice Topográfico						
1	1	0	1	100 %	0 %	**Aprovado**
2	0	0	0			
3	0	0	0			
4	4	1	5	80 %	20 %	
5	299	134	433	69,05%	30,95 %	
Litologia						
1	0	0	0			**Reprovado**
2	42	1	43	97,67 %	2,33 %	
3	0	0	0			
4	0	0	0			
5	4	1	5	80 %	20 5	
6	7	0	7	100 %	0 %	
7	7	1	8	87,50 %	12,50 %	
8	242	129	371	65,23 %	34,77 %	
9	1	0	1	100 %	0 %	
10	0	2	2	0 %	100 %	
11	0	0	0			
12	0	0	0			
13	1	1	2	50 %	50 %	

As primeiras três colunas são escritas pelo programa, as restantes foram criadas manualmente para elucidar o leitor quanto ao processo de análise utilizado para seleccionar a amostra mais conveniente. Em cada classe, contabilizando o número de pontos e calculando a percentagem de pontos atribuídos a cada amostra, verificamos se essa variável teve uma aceitável divisão das observações reais. Assim, se o número total de pontos da amostra de treino, em cada classe, corresponder a um valor próximo de 70% (sobrando 30% para a de validação), esse factor é considerado como aprovado – no caso de existir uma classe com registos na amostra de treino, mas sem instâncias na amostra de validação, o respectivo factor será classificado como reprovado (a menos que uma classe registe apenas uma ocorrência - não é divisível). Entre as várias amostras, aquela que registar mais factores aprovados será a escolhida para dar continuidade ao processo de produção de cartografia de susceptibilidade a movimentos de vertente.

I.4. Incrementação do método do valor informativo

Uma vez definidas as amostras de treino e de validação, estamos em condições de aplicar o método do VI, tendo em vista a criação de um cartograma que expresse a maior ou menor predisposição para ocorrência de movimentos de vertente em cada célula da área de estudo.

Este método estatístico tem sido largamente utilizado em Portugal (Piedade *et al.*, 2011; Garcia, 2012; Cordeiro & Guimarães, 2013; Epifânio *et al.*, 2014, entre muitos outros), tendo vindo a produzir resultados muito satisfatórios na região a Norte de Lisboa e na região do Oeste (Garcia, 2012). De acordo com este último autor, trata-se de um procedimento estatístico de ordem 2 de complexidade derivado de uma simplificação da teoria da probabilidade Bayesiana, que combina de forma bivariada a distribuição espacial dos deslizamentos (variável dependente) com cada uma das variáveis (classes) de cada factor de predisposição (variáveis independentes), ponderando a sua importância com base na respectiva densidade de instabilidades, segundo uma estimativa da susceptibilidade em 2 etapas:

i. Cálculo do peso de cada variável de cada factor de predisposição (Ii), que resulta do seu cruzamento com os movimentos de vertente (VI da classe):

$$Ii = \ln \frac{S_i/N_i}{S/N}$$

Sendo:

Ii – valor informativo da variável *i* (classe *x* do factor de predisposição *y*);

S_i – número de unidades de terreno com deslizamentos e presença da variável *i*;

N_i – número de unidades do terreno com a presença da variável *i*;

S – total de unidades de terreno com deslizamentos;

S – total de unidades de terreno da área de estudo.

i. Estimativa da susceptibilidade para cada unidade de terreno, obtida através do somatório dos VI de cada variável de cada factor de predisposição presente na célula:

$$Ij = \sum_{i=1}^{n} X_{ji} \times li$$

Sendo:

Ij – valor informativo da unidade de terreno;

n – número de variáveis;

X_{ji} – presença (1) ou ausência (0) da variável na unidade de terreno;

li – valor informativo da variável.

Podemos dizer, de modo sucinto, que segundo Garcia (2012), o VI é um *score* que resulta da relação entre o que se pode considerar a densidade média espectável e a probabilidade de ocorrência de movimentos quando condicionada pela presença de um factor. Na prática, este indicador compara a densidade de deslizamentos existente em cada classe do factor de predisposição com a densidade média da área de estudo. Assim, os resultados finais exprimem o posicionamento da variável, ou unidade de terreno, em relação à densidade média. A variável é tanto mais importante para o condicionamento dos movimentos quanto maior for a sua distância ao valor zero. Um valor nulo indica que a variável em causa tem uma densidade de movimento idêntica à densidade média registada, pelo que a sua relação com a instabilidade é inconclusiva. As variáveis que têm uma densidade superior à média, indicando maior possibilidade de ocorrência, apresentam *scores* positivos, sucedendo o oposto com as variáveis com densidade inferior à medida que, como tal, terão *scores* com sinal negativo.

1.5. Avaliação da exactidão do modelo

Face ao exposto, qualquer modelo implica a incrementação de uma proposta de avaliação sobre uma fracção da amostra de dados (registos de eventos e ausências) – validação que estabelece o grau de confiança deste através da comparação dos resultados preditos com o conjunto de observações reais, de maneira a avaliar a sua exactidão ou poder preditivo (Rocha, 2012).[133] Como se disse, este processo deve ser baseado em conjuntos de dados independentes que permitam calcular um indicador para os erros de Tipo I (falsos positivos) e de Tipo II (falsos negativos), o que implicou, previamente, a divisão dos dados de ocorrência em dois subconjuntos, um para treino (70%), outro para teste (30%) (Garcia, 2012; Rodrigues, 2013).

Discriminamos duas formas de avaliação:

i. Comparação directa através da contagem do número de pontos inclusos em cada classe de probabilidade, em que a exactidão será tanto maior quanto maior for o somatório de ocorrências dentro dos limites da classe indicativa de uma probabilidade de acontecimento elevada.

[133] Uma breve nota para falar da distinção entre os conceitos de "exactidão" e "precisão". A precisão diz respeito à aproximação entre os resultados obtidos pela aplicação do procedimento experimental por diversas vezes em determinadas condições (grau de variância em torno da média). Por sua vez, a exactidão assume-se como o grau de aproximação do valor calculado ou estimado relativamente ao valor definido ou real.

ii. Cálculo das medidas de exactidão depois da transformação das predições do modelo (escala contínua) numa matriz binária, alteração que implica a definição (crítica) de um limiar de corte (*threshold*) – de modo a aferir a frequência das distribuições para os grupos dos falsos negativos (FN) e positivos (FP) e verdadeiros negativos (VN) e positivos (VP) (Figura 49) -, e um cálculo com recurso a uma tabela de contingência ou matriz de confusão esquemática, como a apresentada na Tabela XXV (Begueria, 2006; Rodrigues, 2013).

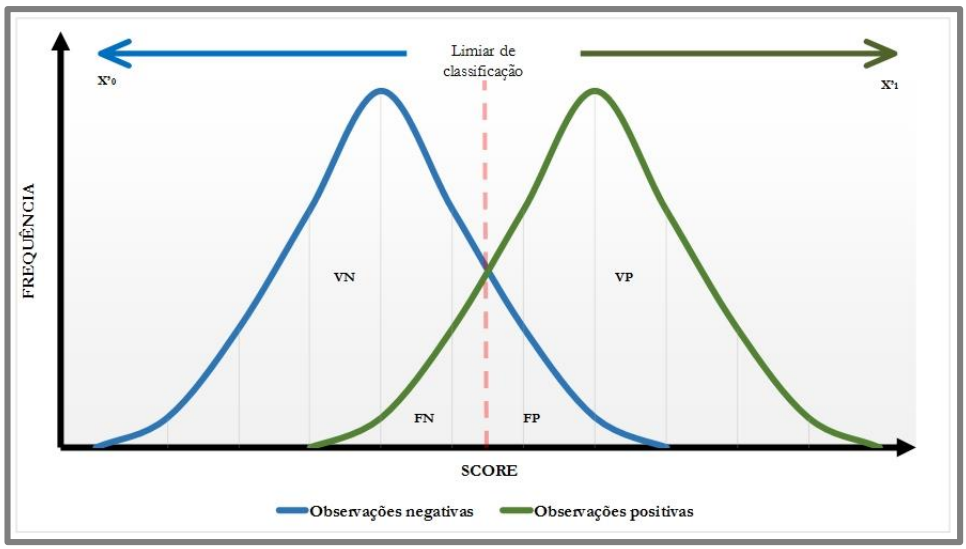


Figura 49 - Frequência das distribuições para os grupos negativos e positivos e o papel do limiar de predição. Adaptado de Begueria (2006).

No presente exercício dedicamo-nos somente ao segundo tipo, em que, numa situação ideal, com uma perfeita discriminação entre as observações positivas e negativas, as distribuições de frequências apareceriam separada no gráfico; porém, na maior parte dos casos, irá ocorrer uma certa sobreposição (Figura 49), levando a erros de predição, representados na matriz como elementos fora da diagonal (Tabela XXV), que, em algumas circunstâncias são designados por erros de comissão e erros de omissão (Rocha, 2012)[134].

Tabela XXV - Matriz de confusão esquemática. Fonte: Rocha (2012), p. 758

	Casos Observados	
Casos Preditos	X_1	X_0
X'_1	VP	FP
X'_0	FN	VN

[134] Os erros de comissão ocorrem quando se atribui uma célula a uma classe à qual ela não pertence (inclusão) – elementos não diagonais de cada linha). Os erros de omissão são traduzidos pela não atribuição, a determinada classe, de um pixel que lhe pertence (exclusão) – elementos não diagonais de cada coluna).

A *Receiver Operating Characteristics* (ROC), baseada nestes princípios, é amplamente utilizada nas mais variadas áreas da ciência (Begueria, 2006; Philips *et al.*, 2006; Garcia, 2012; Chesney *et al.*, 2013; Coelho *et al.*, 2013; Fernández *et al.*, 2013; Rodrigues, 2013). É uma técnica muito útil para visualização, organização e selecção de modos de classificação com base no seu desempenho, tendo a vantagem de, ao contrário de outras como a *Overall prediction success*, *Cohen's Kappa*, *Odds ratio*, *True Skill Statistic*, ser independente do limiar de corte (ou seja, considera todos os limiares possíveis), o que lhe confere maior robustez (Rocha, 2012; Rodrigues, 2013).

A referida técnica de análise realiza-se através da construção e interpretação do chamado gráfico ROC, grafismo bidimensional, nos quais, a taxa de VP é representada no eixo Y e a taxa de FP no eixo X. A taxa de VN surge no eixo do lado oposto ao eixo X e a taxa de FN aparece no eixo oposto ao eixo Y (Figura 50).

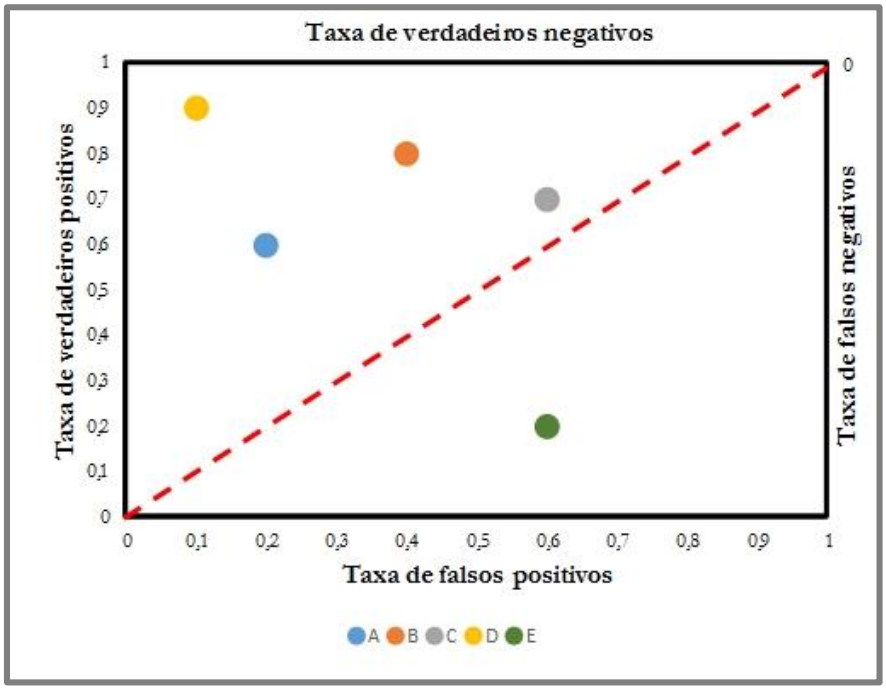


Figura 50 - Gráfico ROC básico mostrando 5 classificadores discretos. Fonte: Rocha (2012), p. 759.

A interpretação do gráfico ROC resume-se ao significado dos seus quatro cantos. O ponto do canto inferior esquerdo (0,0) diz-nos que aquele classificador não comete erros FP, mas também que não ganha verdadeiros positivos. Em sentido oposto, o ponto superior direito (1,1) classifica tudo como observações positivas, a Taxa de verdadeiros positivos é máxima, todavia, o mesmo acontece com a Taxa de falsos positivos. Assim, o ponto (0,1) representa uma classificação perfeita; na Figura 50, o classificador D é o mais próximo dessa conotação (Rocha, 2012). Deste modo,

quanto maior for a diferença entre a curva obtida e a diagonal (que indica um acerto casuístico do modelo) melhor é o ajuste ou capacidade preditiva do modelo (Garcia, 2012).

Segundo Rocha (2012), podemos assumir que a linha diagonal representa a estratégia de interpretar/adivinhar aleatoriamente uma classe. Por exemplo, se um classificador interpreta aleatoriamente a classe positiva metade das vezes, espera-se que ele obtenha correctamente metade dos positivos e metade dos negativos. Então, qualquer classificador que apareça no triângulo inferior direito tem um desempenho pior que a interpretação aleatória.

A interpretação do gráfico ROC poderá ser completada com um indicador acessório, uma medida estatística global do modelo. Para comparar classificadores é possível reduzir o desempenho ou habilidade preditiva de um modelo a um único valor escalar, independentemente do número de registos de ocorrências e não ocorrências (Philips *et al.*, 2006; Rodrigues, 2013). A *Area Under Curve* (AUC) é um valor uni-escalar e é interpretada como a probabilidade de o modelo conseguir prever com exactidão futuras ocorrências do fenómeno, uma vez que afere se a classificação do modelo estatístico em uso produziu um *output* cujos valores apontados como elevadas probabilidades de ocorrência são, de facto, lugares onde já se registou o fenómeno, e vice-versa (Garcia, 2012; Chesney *et al.*, 2013). Os seus valores estão compreendidos entre 0 e 1, quando mais se aproximar da unidade maior será a sua capacidade preditiva. No entanto, na linha do que se disse anteriormente, porque a interpretação aleatória produz uma linha diagonal (cfr. Figura 50), a qual tem uma área de 0,5, nenhum classificador deverá ter uma AUC inferior a esse valor, pois quanto mais distante a curva ROC estiver da linha recta diagonal, melhor o modelo discrimina entre ausências e presenças (Garcia, 2012; Rocha, 2012; Chesney *et al.*, 2013; Rodrigues, 2013).[135]

A operacionalização desta metodologia de validação dos modelos espaciais com finalidades preditivas só pode ser completada com recurso a *software* capaz de executar ferramentas de processamento estatístico de dados alfanuméricos - neste momento, apenas temos conhecimento de uma solução SIG com capacidade para proceder a este tipo de avaliação. Isto dá o mote para um teste particular que coloca em confronto dois dos *software* de estatística mais populares e documentados, o SPSS *Statistics* (SP da IBM) e o R *Statistics*, e a *toolbox* Sextante (incorporada no gvSIG), contrapondo essencialmente a simplicidade e usabilidade de uns e as barreiras de implementação do outro. Enquanto no primeiro (Figura 51) e no Sextante basta correr uma

[135] Garcia (2012), Rocha (2012) e Rodrigues (2013) atribuem as seguintes conotações a cada intervalo de AUC: 0,5 a 0,6 – falha; entre 0,6 – 0,7 – fraca; de 0,7 a 0,8 – razoável; 0,8 a 0,9 – boa; acima disto – excelente.

ferramenta, definindo os seus parâmetros de entrada, no segundo é necessário instalar módulos adicionais e escrever um *script* que execute a análise ROC (Programa 18[136]).

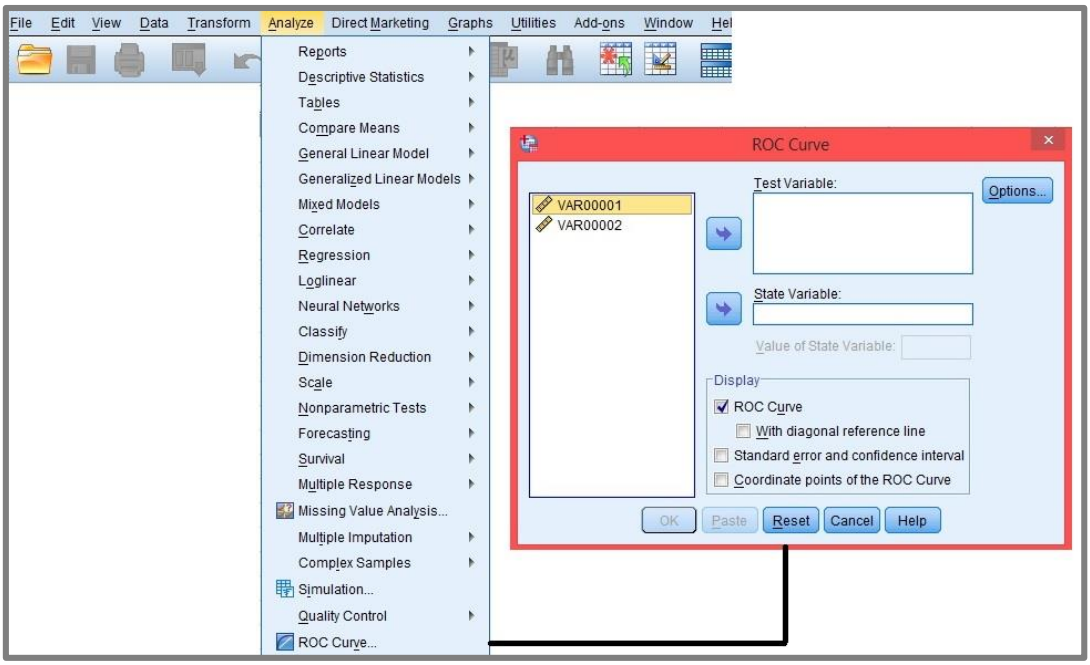


Figura 51 - Procedimento, em IBM SPSS *Statistics* 21, que estabelece o gráfico ROC e calcula o valor de AUC.

A acrescentar a esta avaliação qualitativa (que será motivo de discussão mais adiante), a construção do gráfico ROC e cálculo da AUC motivaram a construção de outro algoritmo para comparação dos vários *software* SIG considerados. Recordamos o procedimento que visava a obtenção de uma amostra de treino e outra de validação. A amostra de validação derivada é constituída apenas por ocorrências, mas, como se esclareceu, a técnica de avaliação descrita tem também como parâmetro de entrada o registo das ausências. Portanto, antes de se usar o *software* de estatística para o cálculo dos indicadores de exactidão do modelo, em ambiente SIG, tem de se incrementar um algoritmo que cria e adiciona as ausências à amostra das presenças, associando cada ponto ao valor de susceptibilidade predito pelo modelo registado nesse local.

[136] **Ver Anexo A, p. A16.**

2. Base analítica para selecção de *software* com funcionalidades para implementação do método do valor informativo, e construção e caracterização dos procedimentos destinados à realização das etapas estabelecidas

> *"O desenvolvimento de Sistemas de Informação Geográfica tornou-se mais muito mais simples do que antes, isto promoveu o desenvolvimento de SIG de código aberto. A lista discutida aqui apenas reflecte a fotografia instantânea tirada neste momento, pois esta imagem pode mudar significativamente no futuro"*
>
> Daoyi Chen; Shahriar Shams; César Carmona-Moreno & Andrea Leone (2010:256)[137]

2.1. Critérios que definem as soluções SIG a serem testadas

Com o intuito de compor uma AMC organizada segundo um conjunto de critérios que nos parecem razoáveis, tendo em linha de conta os objectivos que nos propomos atingir, importa, em primeiro lugar, achar 10 soluções SIG que integrem essa avaliação multicritério. Como se ilustrou na Figura 44 (cfr. p. 152), a macro disponibilizada na página da *Business Perfomance Managment* viabiliza a comparação entre dez alternativas, ou seja, dos elementos da Tabela IX (cfr. p. 54) temos de seleccionar as 10 alternativas mais promissoras.

Esta pré-selecção, apresentada na Tabela XXVI, resulta essencialmente da avaliação de três parâmetros: por um lado, procuramos *software* referidos na bibliografia como soluções de confiança e de alta rentabilidade; por outro, valorizamos *software* com os quais já estamos familiarizados (contacto em experiências passadas) – com o conhecimento prévio sobre as funcionalidades do *software*, torna-se mais fácil de operar e construir rotinas perfeitamente ajustadas e adaptadas ao problema. Por fim, para uma solução SIG livre e/ou de código aberto ser considerada, a mesma

[137] No original, *"As the GIS development became much simpler than before, it had promoted the development of open source GIS and a number of projects are still undergoing. The list to be discussed here only reflects a snapshot at the current time, and the picture may change remarkably in the future*" *In* CHEN, Daoyi; SHAMS, Shahriar; CARMONA-MORENO, César & LEONE, Andrea (2010) – "Assessment of open source GIS *software* for water resources management in developing countries". **Journal of Hydro-environment Research**, N.º 4, 253-264.

deve ter uma distribuição para Microsoft Windows, pois, tal como se disse inicialmente na parte deste trabalho referente aos aspectos matriciais da metodologia utilizada, a solução proprietária de referência só funciona neste sistema operativo (elemento neutral na realização dos testes – isto é, garante as mesmas condições de teste).

Tabela XXVI - *Software* pré-seleccionado para posterior avaliação de predisposição para a concretização da meta estabelecida

	***Software* pré-seleccionado:**	**Razões para eleição:**
0	GDAL/OGR	-» Não é um SIG *desktop*, mas julgamos que tem as ferramentas necessárias para executar a maior parte das tarefas que compõem este exercício; -» Experiência de utilização dos seus algoritmos.
1	GRASS GIS	-» Inúmeras aplicações práticas em vários ramos científicos (https://grass.osgeo.org/documentation/applications/, acedido 3 Novembro, 2015); -» Considerado como *software* maduro por Steiniger & Hunter (2013); -» Bons resultados obtidos no ensaio de Akbari & Rajabi (2013); -» Cotação de 81% na classificação estabelecida no âmbito Projecto CASCADOSS.
2	gvSIG	-» Inúmeras aplicações práticas (http://www.gvsig.com/pt/difusao/artigos, acedido em 3 Novembro, 2015); -» Considerado como *software* maduro por Steiniger & Hunter (2013); -» Cotação de 70% na classificação estabelecida no âmbito Projecto CASCADOSS; -» Entre todas as soluções avaliadas por Chen *et al.* (2010), o gvSIG faz parte do *top 4*. -» Experiência de utilização das suas ferramentas.
3	ILWIS	-» Considerado como *software* maduro por Steiniger & Hunter (2013).
4	Map Window GIS	-» Entre todas as soluções avaliadas por Chen *et al.* (2010), o Map Window faz parte do *top 4*. -» Considerado como *software* maduro por Steiniger & Hunter (2013); -» Outros trabalhos como o de Lei *et al.* (2011).
5	OpenJUMP	-» Experiência de utilização das suas ferramentas; -» Considerado como *software* maduro por Steiniger & Hunter (2013); -» Entre todas as soluções avaliadas por Chen *et al.* (2010), o OpenJUMP faz parte do *top 4*.
6	QGIS	-» Experiência de utilização das suas ferramentas; -» Entre todas as soluções avaliadas por Chen *et al.* (2010), o QGIS faz parte do *top 4;* -» Considerado como *software* maduro por Steiniger & Hunter (2013); -» Cotação de 82% na classificação estabelecida no âmbito Projecto CASCADOSS – *software* com maior cotação.
7	SAGA GIS	-» Resultados razoáveis obtidos no ensaio de Akbari & Rajabi (2013); -» Considerado como *software* maduro por Steiniger & Hunter (2013);

		-» Inúmeras aplicações práticas em vários ramos científicos (http://www.saga-gis.org/en/index.html, acedido 3 Novembro, 2015).
8	Sextante incorporado no gvSIG	-» *Toolbox* que disponibiliza uma ampla gama de bons algoritmos de geoprocessamento; -» Incorporável em *software* como o gvSIG, QGIS, OpenJUMP ou ArcGIS.
9	SPRING	-» Experiência de utilização das suas ferramentas.

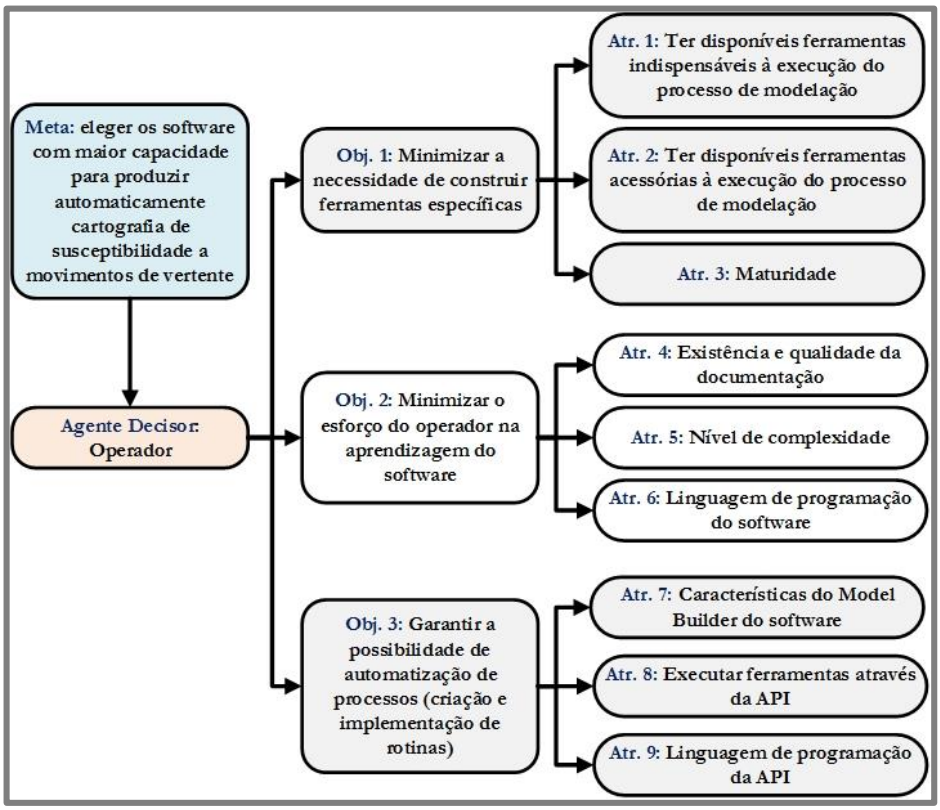


Figura 52 - Estrutura da AMC que tem como desiderato a eleição de *software* para produção automática de cartografia de susceptibilidade a movimentos de vertente – definição da meta, agentes decisores, objectivos e atributos.

Os 10 *software* pré-seleccionados foram valorados quanto aos vários atributos que constam na Figura 52. A estrutura da AMC esquematizada nesta ilustração adapta-se aos objectivos e conhecimentos do operador que concebeu esta dissertação, pois, tal como se referiu anteriormente, qualquer cadeia de operações que vise a escolha de algo segundo um conjunto de critérios está dependente de um ou mais agentes decisores, que, em determinado contexto (neste caso, uma dissertação de mestrado onde se compara *software*), possuem preferências e delimitam os objectivos, atributos e a orientação destes últimos aos anteriores, segundo aquilo que lhes parece ser mais ajustado à meta naquela conjuntura. Essencialmente, isto serve para dizer que não existem modelos singulares, logo, cada organismo da administração pública tem de ser capaz de formular as suas

próprias estratégias de decisão, visto que só os seus funcionários conhecem com rigor a realidade e contexto da instituição onde se inserem.

A Tabela XXVII apresenta os resultados da valoração dos vários *software* relativamente aos vários atributos em consideração, fruto da aplicação do PAH – em todas as valorações, certificámo-nos que a razão de consistência (RC) estava abaixo dos 10 % (Saaty, 1988).

Tabela XXVII - Resultados do processo de valoração das várias alternativas em cada atributo considerado, segundo o PAH

Soft.	Atr.1	Atr.2	Atr.3	Atr.4	Atr.5	Atr.6	Atr.7	Atr.8	Atr.9
0	0,08	0,04	0,25	0,25	0,02	0,18	0,02	0,14	0,26
1	0,29	0,35	0,25	0,25	0,23	0,18	0,21	0,14	0,26
2	0,04	0,09	0,08	0,04	0,23	0,02	0,21	0,14	0,03
3	0,13	0,04	0,03	0,04	0,05	0,02	0,02	0,02	0,03
4	0,03	0,02	0,03	0,04	0,05	0,18	0,02	0,02	0,03
5	0,04	0,09	0,03	0,02	0,05	0,02	0,02	0,02	0,03
6	0,08	0,09	0,25	0,025	0,23	0,18	0,21	0,14	0,26
7	0,08	0,09	0,03	0,04	0,05	0,18	0,02	0,14	0,03
8	0,21	0,16	0,03	0,04	0,04	0,02	0,21	0,14	0,03
9	0,03	0,02	0,03	0,02	0,05	0,02	0,02	0,14	0,03
	1	1	1	1	1	1	1	1	1
RC	1%	2%	1%	2%	2%	0%	0%	0%	0%

A atribuição de um valor numérico a cada uma das alternativas em todos os atributos é um processo subjectivo e relativo ao contexto da realização dos exercícios de desempenho. Aqui valorizaram-se três aspectos: i) economização de tempo, evitando a necessidade de construção das nossas próprias ferramentas; ii) considerando os limitados conhecimentos de que dispúnhamos sobre como operar a maior parte das soluções SIG livres, valorizámos aquelas que tornem a tarefa de aprendizagem mais simples e acessível; iii) uma vez que nos propusemos comparar o tempo que os vários *software* consomem no desempenho de uma cadeia sequenciada de tarefas de geoprocessamento, é relevante automatizar os procedimentos, facilitando a contabilização desse tempo.

Por outro lado, a valoração está condicionada pela nossa interpretação dos vários atributos. O primeiro e segundo atributo são os mais objectivos. Definimos *a priori* o esquema mental dos algoritmos que pretendemos implementar nestes exercícios, portanto sabemos de que ferramentas necessitamos (Tabela XXVIII), umas são indispensáveis (não há alternativas, têm de ser usadas aquelas), outras não o são, isto é, não é estritamente necessário usar-se aquela ferramenta, pois, com a adaptação do raciocínio, o emprego de outra pode conduzir ao *output* pretendido. No que diz respeito a este atributo, o GRASS GIS é claramente o pacote que dispõe de uma biblioteca (repositório) de algoritmos mais rica, que inclui todos aqueles de que necessitámos. Não obstante,

não se pode esquecer a *toolbox* Sextante, que conta também com a presença da maior parte desses algoritmos, dispondo também de uma extensa variedade de ferramentas destinadas a resolver problemas avançados.

Tabela XXVIII - Ferramentas que os *software* a serem seleccionados devem ter para executar o procedimento de produção de cartografia de susceptibilidade a movimentos de vertente

Ferramentas indispensáveis:	Ferramentas acessórias:
-» Leitura de ficheiros vectoriais;	-» Agregação (*dissolve*);
-» Leitura de ficheiros matriciais;	-» *Split;*
-» Conversão *Raster To Polygon*;	-» *Merge;*
-» Conversão *Polygon To Raster*;	-» *Select > Export;*
-» Intersecção;	-» *Extract*;
-» Reclassificação de ficheiros matriciais;	-» Soma ponderada;
-» Calculadora de ficheiros matriciais;	-» *Buffer;*
-» Extracção de valores por localização;	-» União de polígonos;
-» Criação de pontos aleatórios.	-» *Erase*.

Não podemos deixar de realçar o facto de que estas são ferramentas básicas de qualquer *software* SIG, no entanto, algumas delas não estão presentes em algumas distribuições enquanto ferramentas nativas, conotando-as de alguma imaturidade, facto que nos parece, contudo, poder ser relativizado se as virmos como plataformas interoperáveis. Por exemplo, as ferramentas do módulo Sextante podem ser incorporadas no gvSIG, no GRASS ou no QGIS. Este último, por sinal, reúne numa mesma interface uma grande variedade de algoritmos de outros pacotes SIG.

Retomando, ao contrário dos dois primeiros atributos (cfr. Figura 52), o atributo 3, que expressa o grau de maturidade das alternativas consideradas, é bastante mais subjectivo, na medida em que tomamos como indicador de maturidade a associação do *software* SIG à OSGeo, assumindo que o facto de um *software* ser considerado como projecto OSGeo é sinónimo, tanto de maturidade, como de qualidade. Outros, certamente, considerariam outros modos de avaliar este parâmetro.

Os atributos 4, 5 e 6 são igualmente subjectivos e relativos à interpretação do operador. Por um lado, valorizou-se sobretudo a existência de documentação sobre a construção de *scripts* preparados para correr na API do *software*. Por outro, valorizaram-se alternativas que primam pela simplicidade da sua interface (plataforma *user-friendly*), e neste sentido, a interpretação do nível de complexidade obedece:

i. À predisposição pessoal que cada um tem para interagir com uma interface. Enquanto muitos acham o GRASS uma plataforma pouco amigável e complexa, nós não temos a mesma opinião, inclusive, julgamos que este *software* patenteia simplicidade e tem a

vantagem de ter um manual do utilizador equiparável ao do ArcGIS, tornando o seu manuseamento acessível a qualquer operador SIG com abertura suficiente para abraçar uma identidade e filosofia muito própria.

ii. Ao modo como se considera o *software*. Por exemplo, a biblioteca GDAL/OGR é de simples utilização quando integrado em qualquer *software* SIG, porém quando o usamos em contexto *standalone*, a sua manipulação não é tão simples, o mesmo vale para a *toolbox* Sextante.

Por fim, a linguagem de programação que define a sintaxe do código-fonte é crucial, na medida em que o operador vai preferir as linguagens em que se sente mais confortável. Tratando-se de *software* de código aberto, faz todo o sentido escolher um programa com uma linguagem que sabemos interpretar, possibilitando assim o estudo do código-fonte. Para além disto, é possível que a linguagem influencie a robustez do código, sendo também um parâmetro associável à qualidade. Como se sabe, tanto a linguagem C/C++ como a linguagem Java são as mais usadas na construção de *software* SIG (cfr. Tabela IX, p. 54); trata-se de sintaxes compiladas, portanto, mais eficientes quando comparadas às interpretadas (como o Python). As diferenças entre elas são conhecidas e podem justificar diferenças na eficiência dos algoritmos, basta relembrar que a Java corre em máquina virtual e que a C++ corre directamente na raiz do computador. Como os nossos conhecimentos em C++ e Java são muito superficiais, não estamos capacitados para tirar partido da interpretação do código-fonte; então, na valoração deste atributo, usamos esta última hipótese relacionada com a eficiência das linguagens como elemento diferenciador.

Para concluir a análise, resta-nos esclarecer os últimos três atributos, de cariz objectivo mas cuja inclusão simboliza preciosismos do operador. Já se fez referência ao interesse que tínhamos em automatizar processos; há duas formas de o fazer em ambiente SIG: através da construção de cadeias de processos em ambiente gráfico (interface comummente conhecida como *Model Builder*), ou através da construção de programas que correm na API do *software*. Assim, estes atributos destacam com valores mais elevados as alternativas que dispõem de *Model Builder* (gvSIG, GRASS, QGIS e Sextante) e de API (GDAL/OGR, GRASS, gvSIG, QGIS, SAGA, Sextante, Spring) - cfr Tabela XXVII. Como a linguagem que melhor dominamos é Python, com o último atributo damos primazia às alternativas, cuja API executa programas nesta linguagem.

Pela explicação depreende-se que os atributos não têm o mesmo peso. A Tabela XXIX revela os pesos induzidos a cada atributo, o produto de uma estimativa que colocou na linha da frente aspectos relacionados com a maturidade (relação maturidade/qualidade) e a existência de documentação que exponha técnicas de *scripting* em Python. Em coerência com este último, o

operador privilegiou a existência de uma API em linguagem Python. No mesmo nível de importância encontra-se a disponibilidade de ferramentas consideradas como fundamentais. Aspectos como o nível de complexidade (pela subjectividade), a linguagem de programação do *software* (não dispomos de conhecimentos avançados em C ou Java) e as características do *model builder* (dá-se, antes, preferência às propriedades da API) foram relegados para segundo plano.

Tabela XXIX - *Score* final que indica a predisposição das alternativas relativamente à meta estipulada, tendo em conta os vários critérios e o peso que cada um deles deverá ter na tomada de decisão

Soft.	Atr.1	Atr.2	Atr.3	Atr.4	Atr.5	Atr.6	Atr.7	Atr.8	Atr.9	Score
0	1,12	0,28	6,25	6,25	0,06	0,54	0,06	0,98	3,64	**19,18**
1	4,06	2,45	6,25	6,25	0,69	0,54	0,63	0,98	3,64	**25,49**
2	0,56	0,63	2	1	0,69	0,06	0,63	0,98	0,42	**6,97**
3	1,82	0,28	0,75	1	0,15	0,06	0,06	0,14	0,42	**4,68**
4	0,42	0,14	0,75	1	0,15	0,54	0,06	0,14	0,42	**3,62**
5	0,56	0,63	0,75	0,5	0,15	0,06	0,06	0,14	0,42	**3,27**
6	1,12	0,63	6,25	0,625	0,69	0,54	0,63	0,98	3,64	**15,11**
7	1,12	0,63	0,75	1	0,15	0,54	0,06	0,98	0,42	**5,65**
8	2,94	1,12	0,75	1	0,12	0,06	0,63	0,98	0,42	**8,02**
9	0,42	0,14	0,75	0,5	0,15	0,06	0,06	0,98	0,42	**3,48**
Peso	**0,14**	**0,07**	**0,25**	**0,25**	**0,03**	**0,03**	**0,03**	**0,07**	**0,14**	

Neste quadro, os valores associados a cada alternativa resultam da multiplicação dos valores da Tabela XXVII por 100 e pelo respectivo peso que cada atributo tem. A soma destes valores ponderados realça os *software* melhor cotados: a biblioteca GDAL/OGR, o GRASS GIS e o QGIS. Por serem aquelas que se destacam de modo evidente em relação às outras aplicações, estas são as alternativas que passam à fase seguinte.

2.2. Características dos dados de entrada e propriedades das rotinas que implementam os algoritmos de cálculo da susceptibilidade a movimentos de vertente, e características dos dados de entrada

A dificuldade com que cada *software* desempenha uma tarefa é necessariamente condicionada pelas características dos dados de entrada. Em teoria, quanto maior for o volume de dados, menor será a eficiência das ferramentas. Este é o motivo de estudo da escalabilidade. No presente exercício, não vamos ter em conta a variação da performance com o volume de dados, ainda assim, sabendo que todos os procedimentos têm os mesmos dados de entrada, importa identificar algumas propriedades relativas a estes *inputs,* elementos cruciais para explicar alguns dos resultados obtidos.

A Tabela XXX expõe as propriedades que nos pareceram mais relevantes. Quando se falou na preparação dos dados, explicou-se que a variável dependente foi preparada em formato vectorial; por seu turno, as variáveis independentes foram categorizadas em formato matricial. Não obstante, ao longo dos vários procedimentos foram realizadas várias conversões (incontornáveis), daí termos optado por incluir o número de *features*/células e os *Kilobytes* (KB) ocupados em disco pelos temas em ambos os formatos.

Tabela XXX - Características dos dados de entrada do exercício de produção de cartografia de susceptibilidade

		Vectorial - *.shp*		Matricial - *.tif*	
Tema	**N.º Categorias**	**N.º *Features***	**KB**	**N.º Células**	**KB**
Movimentos (pontos)		441	13		
Movimentos (polígonos)		441	280	1368	34
COS	12	95	763	1 600 000	62
Curvatura	5	143 669	38 512	1 600 000	316
Declives	9	44 830	16 046	1 600 000	257
Exposições	9	23 896	10 946	1 600 000	324
Inverso índice topográfico	5	35 472	7 167	1 600 000	208
Litologia	13	146	533	1600000	50

A análise dos resultados permite verificar que se trata de dados que ocupam pouca memória em disco, mas que podem ser um desafio para qualquer *software* se considerarmos o número de *features* que resultam da conversão de *raster* para vector, de alguns factores de predisposição, nomeadamente, a curvatura das vertentes.

Resta, por fim, apresentar e caracterizar os vários programas que concretizam os algoritmos conceptuais referidos no primeiro tópico deste capítulo nos diferentes *software* considerados (ArcGIS 10.2; GRASS GIS 7.0.0, QGIS 2.10.1 e GDAL/OGR 1.11.1), explicitando genericamente a cadeia de processos lógicos que implementam e as suas principais propriedades (Tabela XXXI):

i. **Avaliação da Independência Condicional entre as variáveis independentes (Programas 19[138], 20[139], 21[140] e 22[141]) -»** A extensão SDM - http://www.ige.unicamp.br/sdm/ (acedido em 19 Setembro, 2015) – fornece um *script*

[138] **Ver Anexo A, p. A17.**
[139] **Ver Anexo A, p. A19.**
[140] **Ver Anexo A, p. A22.**
[141] **Ver Anexo A, p. A25.**

que calcula o RIC e TAC. Tratando-se de uma *toolbox* de código aberto, foi possível adaptar o código original (executável apenas na versão 9.3 do ArcGIS), de modo a ser implementado no ArcGIS 10.2, GRASS GIS 7.0.0, QGIS 2.10.1 e GDAL 1.11.1. As operações realizadas por estes programas são triviais – trata-se de instruções que visam a obtenção de propriedades de cada uma das imagens (resolução da célula, número total de células e número de células por classe) e a iteração das *features* que representam os vários movimentos. Segundo Bonham-Carter (1994) e Agterberg & Cheng (2002), estes são os elementos necessários para o cálculo dos indicadores referidos, cujas fórmulas se encontram nos diversos programas, constando também no *script* original da SDM.

ii. **Criação de amostras ideais (Programas 23[142], 24[143], 25[144] e 26[145]) -»** Ao contrário do ponto anterior, em que se avalia sobretudo a capacidade de leitura de dados geográficos (vectoriais e matriciais), os programas usados para a criação de amostras de ajuste e de validação do modelo manipulam os dados geográficos através da incrementação de geoprocessos simples, frequentemente utilizados e transversais à maior parte dos *software* SIG. São cadeias de decisões que estabelecem, interactivamente e a partir da cisão do inventário de ocorrência inicial, dois conjuntos de dados independentes, uma amostra de treino (70 %) e outra de validação (30 %) para cada factor condicionante do fenómeno, segundo a seguinte sequência lógica de etapas:

   a. Conversão do factor, que se encontra em formato de dados matricial, para formato de dados vectorial;

   b. Organização das classes de cada factor em ficheiros (*shapefile*) individuais;

   c. Intersecção de cada uma das classes com o tema pontual que identifica as presenças;

   d. Quantificação do número de pontos resultante dessa operação, e divisão dos mesmos em dois novos temas, um com 70% do número total de pontos registados e outro com 30%.

[142] **Ver Anexo A, p. A27.**
[143] **Ver Anexo A, p. A30.**
[144] **Ver Anexo A, p. A34.**
[145] **Ver Anexo A, p. A37.**

Tabela XXXI - Algumas características dos programas construídos no âmbito do exercício de produção de cartografia de susceptibilidade

Software	Programa.	Linhas	N.º Geoprocessos	N.º Ferramentas nativas	N.º Ferramentas de outras distribuições	N.º Ferramentas criadas	*Features*/Células processadas	Outputs intermédios (Kb)	Complexidade
Avaliação da Independência Condicional entre as variáveis independentes									
ArcGIS 10.2	19	137	2	2	0	0	441 *features;* 9 600 000 células	0	1
GRASS GIS 7.0.0	20	168	4	4	0	0	882 *features* 19 200 000 células	0	1
QGIS 2.10.1	21	170	1	1	1	0	441 *features;* 9 600 000 células	0	1
GDAL/OGR 1.11.1	22	147	2	2	0	0	441 *features;* 9 600 000 células	0	1
Criação de amostras ideais									
ArcGIS 10.2	23	182	7	7	0	0	519 663 *features;* 9 600 000 células	143 000	1
GRASS GIS 7.0.0	24	196	6	5	0	1	4 975 831 *features;* 19 200 000 células	635 000	1
QGIS 2.10.1	25	206	4	2	2	0	4 975 390 *features;* 9 600 000 células	143 000	1
GDAL/OGR 1.11.1	26	342	5	2	0	3	4 975 390 *features;* 9 600 000 células	?	3
Incrementação do método do valor informativo									
ArcGIS 10.2	27	139	8	8	0	0	499 409 *features*; 32 027 000 células	182 000	1
GRASS GIS 7.0.0	28	177	10	10	0	0	746 796 *features*; 41 627 000 células	904 000	1
QGIS 2.10.1	29	157	6	2	4	0	746 487 *features;* 32 027 000 células	?	1
GDAL/OGR 1.11.1	30	270	7	5	0	2	746 487 *features;* 32 027 000 células	?	3
Avaliação da exactidão do modelo									
ArcGIS 10.2	31	43	7	7	0	0	1015 *features;* 1 600 000 células	660	1
GRASS GIS 7.0.0	32	105	11	11	0	0	2519 *features;* 22 400 000 células	3 264	2
QGIS 2.10.1	33	70	6	4	1	1	1015 *features;* 1 600 000 células	660	1
GDAL/OGR 1.11.1	34	91	3	0	0	3	? *features* 1 600 000 células	0	3

e. Uma vez efectuada a intersecção e divisão para todas as classes do factor, as amostras que correspondem aos 70% de ocorrências são anexadas num único tema vectorial (tarefa de *merge* que dá origem a uma amostra de treino com repartição ideal do número de pontos por cada classe nesse factor) – é feito o mesmo para as amostras com 30% das ocorrências em cada categoria.

f. Possuindo uma amostra de validação e de treino para cada um dos factores independentes, e sabendo que cada uma delas corresponde à repartição (70%-30% dos pontos naquela variável, falta avaliar se a divisão entre as amostras respeita (ou não) essa regra de cisão nos outros factores, de maneira a garantir que todas as classes (de todos os factores) onde se registam presenças têm representatividade aceitável nos dois conjuntos. Este escrutínio implicou que para cada uma das amostras geradas, fossem intersectadas as primeiras com cada classe de cada um dos factores; contabilizou-se o número de pontos presentes no resultado e registaram-se esses valores numa folha de cálculo (cfr. Tabela XXIV).

iii. **Incrementação do método do VI (Programas 27[146], 28[147], 29[148] e 30[149]) -»** Depois de escolher a amostra que garante a expressão de todas as classes (ou pelo menos da maior parte), esta foi usada, juntamente com os seis factores de predisposição, como tema de entrada de um conjunto de programas que implementam o método do VI, seguindo as seguintes etapas:

a. Obtenção da resolução da célula dos vários factores independentes;

b. Cálculo do número total de pixéis que constituem a área de estudo;

c. Cálculo do número de pixéis onde se registam presenças;

d. Conversão dos temas relativos a cada factor para formato vectorial, e intersecção das ocorrências (tema poligonal) com cada classe para contabilização do número de pixéis que representam a ocorrência do fenómeno nesta classe;

e. Contabilização do número de pixéis que compõem cada classe de todos os factores de predisposição, utilizando para o efeito a tabela *dbf* que acompanha os ficheiros matriciais.

[146] **Ver Anexo A, p. A43.**
[147] **Ver Anexo A, p. A45.**
[148] **Ver Anexo A, p. A49.**
[149] **Ver Anexo A, p. A51.**

f. Cálculo do VI relativo a cada classe através da conjugação, segundo o enunciado da fórmula anteriormente apresentada, entre o número total de pixéis da área de estudo, o número total de pixéis que representam as presenças, o número de pixéis de cada classe e o número de pixéis que evidenciam ocorrência naquela classe.

g. *Lookup* ou reclassificação das matrizes originais, criando-se novos ficheiros em que o valor que identifica a classe é substituído pelo respectivo valor informativo.

h. Somatório destas grelhas, tendo em vista a obtenção de uma matriz que represente a susceptibilidade de cada célula relativa à ocorrência de movimentos de vertente.

iv. **Avaliação da exactidão global do modelo (Programa 31[150], 32[151], 33[152] e 34[153]) -»** A avaliação ROC só pode ser completada com uma amostra que integre tanto presenças como ausências. Assim, estes dois programas prevêem a geração de um conjunto de pontos (não coincidentes com as presenças) que representam a não ocorrência do fenómeno, bem como a extracção dos valores que representam a susceptibilidade nesses lugares, constituindo assim uma tabela que será usada pelo *software* de estatística específico para construção do gráfico ROC e cálculo da AUC. A gestação desta tabela processa-se da seguinte maneira:

a. Contabilizou-se o número de pontos que constituem a amostra de validação;

b. Gerou-se uma circunferência em torno das várias presenças que constituem a amostra de validação;

c. Utilizou-se este *buffer* para apagar os lugares limítrofes às ocorrências no tema que representa os limites da área de estudo;

d. Procedeu-se à criação, em número igual às presenças que constituem a amostra de validação, de pontos aleatórios na área onde se verifica que não existem quaisquer registos de presenças;

e. Procedeu-se à extracção dos valores das células do *raster* de susceptibilidade que se cruzam com as presenças e ausências, guardando essa informação na tabela de atributos desse tema pontual.

[150] **Ver Anexo A, p. A56.**
[151] **Ver Anexo A, p. A57.**
[152] **Ver Anexo A, p. A59.**
[153] **Ver Anexo A, p. A61.**

3. Comparação de desempenho entre *Software* SIG Proprietário e *Software* SIG Livre: 1ª Análise

> *"O QGIS… rivaliza com os mais caros software proprietários, tanto em termos de funcionalidades como de usabilidade. É também um modelo da melhor tecnologia de código aberto geoespacial disponível."*
>
> Joel Lawhead (2015:vii)[154]

No âmbito da produção de cartografia de susceptibilidade a movimentos de vertente realizaram-se quatro testes, dos quais resultam reflexões relevantes sobre o desempenho dos *software* considerados - ArcGIS 10.2, GRASS GIS 7.0.0, QGIS 2.10.1 e GDAL 1.11.1.

3.1. Averiguação da Independência Condicional entre as variáveis independentes

A execução dos Programas 19[155], 20[156], 21[157] e 22[158] revelam uma clara desvantagem do SL/CA relativamente à alternativa proprietária, no que diz respeito à leitura de uma *shapefile* de pontos com 441 instâncias, e de seis imagens matriciais com 1 600 000 células cada. De acordo com a Tabela XXXII, o ArcGIS necessita apenas de 2,08 segundos (0,85% do tempo máximo registado). O GRASS GIS é, de todos, o mais lento, precisando de 4 minutos. A GDAL/OGR e o QGIS apresentam valores muito semelhantes, isto porque o segundo não dispõe, até onde nos foi possível perceber, de um modo de leitura das várias células de um *raster* sem recorrer à biblioteca GDAL. Portanto, no que concerne ao QGIS estamos, neste caso, a avaliar a capacidade de integração da biblioteca GDAL/OGR, que, aqui, revela resultados muito satisfatórios, pois supera a velocidade

[154] No original, *"The open source geographic information system, QGIS, at version 2.6 now rivals even the most expensive commercial GIS software in both functionality and usability. It is also a showcase of the best geospatial open source technology available."* In LAWHEAD, Joel (2015) – **QGIS Python Programming Cookbook**. Packt Publishing, 1ª Edição, Birmingham, 315 p.

[155] **Ver Anexo A, A17.**

[156] **Ver Anexo A, A19.**

[157] **Ver Anexo A, A22.**

[158] **Ver Anexo A, A25.**

da GDAL/OGR quando usado de forma isolada, o que até pode indiciar algum esforço de optimização da rentabilidade de ferramentas externas aquando da sua integração no QGIS.

Tabela XXXII - Tempo consumido na execução da tarefa de Cálculo dos Indicadores de Independência Condicional

Software SIG	Horas:Min:Seg:	Minutos	PTRTM[159]
ArcGIS 10.2	00:00:02,08	0,035	0,85 %
GRASS GIS 7.0.0	00:04:05,04	4,084	100 %
QGIS 2.10.1	00:01:37,99	1,633	39,99 %
GDAL 1.11	00:01:42,14	1,702	41,68%

Pensamos que o GRASS destoa dos demais porque processa os dados o dobro das vezes. Os desenvolvedores desta aplicação SIG conceberam e adoptaram uma filosofia muito própria ao seu projecto, acentuando algumas peculiaridades que distinguem o GRASS dos demais. Uma delas tem a ver com o facto de este *software* apenas trabalhar com dados que se encontrem em formato *grass vector* e *raster vector*. Para não inviabilizar a interoperabilidade entre aplicações, os desenvolvedores incluem na biblioteca de algoritmos deste SIG *desktop,* ferramentas que permitem a importação/exportação para formatos suportados pela biblioteca GDAL/OGR. Assim, todos os programas construídos e adaptados à API do GRASS contemplam a conversão de todos os dados de entrada para os seus formatos nativos. Neste caso específico, isto poderá explicar o atraso registado relativamente às outras alternativas.

Independentemente das disparidades de resultados visíveis na tabela anterior, dá-se ênfase ao argumento que nos diz que qualquer exercício deste género é subjectivo e deve ser relativizado, podendo até ser facilmente enviesado. Como o nosso intuito não é colocar as várias alternativas em situações que lhe sejam favoráveis esclarecemos, desde já, os motivos que se encontram por detrás destes resultados.

Na preparação dos programas que conduziram às instâncias acima descritas tomou-se por princípio que a iteração pelas células das imagens fosse feita segundo os modos normais de leitura das mesmas (vulgo "carregar *layer*"). Isto dá clara vantagem ao *software* da ESRI, na medida em que, este, ao adicionar uma *raster layer* à *view*, concede ao utilizador o acesso à informação tabular (caso ela exista) anexa ao ficheiro matricial, que compila os dados necessários para o cálculo da independência condicional, acelerando significativamente o processo.

[159] Percentagem de tempo despendido relativamente ao tempo máximo registado.

Pelo contrário, os outros *software* não lêem directamente a tabela *.dbf* associada à imagem, portanto, a def *conta_celula*, a def *conta_nrpixeis* e a def *nr_pixel_classe* dos Programas 20, 21 e 22 estão programadas para efectuar a leitura de todas as células, consumindo necessariamente mais tempo, e justificando as discrepâncias evidenciadas na Tabela XXXII.

O utilizador comum, que transita do ArcGIS para uma solução livre, estranharia o facto de, ao carregar a imagem, não ter acesso à tabela, e provavelmente ficaria confuso. Na realidade, o SL/CA obriga a adaptações e estimula a imaginação do operador, fazendo-o equacionar novas abordagens. O operador terá de se aperceber de algumas diferenças de funcionamento, numa tentativa de optimizar soluções. Neste caso nem é assim tão difícil, basta apenas deixar de lado a imagem e considerar apenas a tabela *.dbf*, que disponibiliza a informação necessária.

Assim, no GRASS GIS (Programa 20[160]), substituindo a def *conta_celula* e a def *nr_pixel_classe* pelos métodos[161] detalhados no Programa 35[162], obteremos resultados mais próximos dos da solução proprietária. A def *db_in_ogr* substitui a def *r_in_gdal*, ou seja, em vez de adicionarmos a imagem, fazemos com que o programa apenas considere a tabela; consequentemente, as outras duas têm de ser modificadas de modo a serem capazes de ler a informação tabular, utilizando para o efeito o módulo *sqlite3*[163].

No GDAL (Programa 22[164]), a def *conta_nrpixeis* e def *nr_pixel_classe* foram repostas pelas apresentadas no Programa 36[165] – a biblioteca GDAL/OGR (módulo OGR) tem a capacidade de ler o formato *shapefile* como um todo, ou seja, a geometria e o seu cortejo de ficheiros nucleares – o ficheiro que armazena o directório de dados e a tabela (*.shp, .shx* e *.dbf*, respectivamente) ou individualmente, tornando-se, assim, viável a leitura da tabela do *raster* como se se tratasse de um ficheiro de formato vectorial.

A Tabela XXXIII publicita os novos resultados, que expressam, como seria de esperar, uma redução substancial dos valores relativos ao tempo de desempenho superando, no caso da GDAL/OGR (individualmente ou integrado no QGIS), o ArcGIS. Não podemos deixar de sublinhar este facto que, quando comparado com os resultados plasmados na Tabela XXXII,

[160] **Ver Anexo A, p. A19.**
[161] Em Python, um método ou definição é uma abstracção para uma operação mais ou menos complexa.
[162] **Ver Anexo A, p. A62.**
[163] O SQLite é uma biblioteca com uma licença do tipo "domínio público" que disponibiliza ferramentas para a manipulação e inquirição de dados que constam de tabelas de uma BD, usando a linguagem SQL.
[164] **Ver Anexo A, p. A25.**
[165] **Ver Anexo A, p. A63.**

revelam que o SL/CA pode ser tão ou mais performante que o SP, dependendo muito das competências do operador.

Tabela XXXIII - Tempo consumido na execução da tarefa de Cálculo dos Indicadores de Independência Condicional – adaptações que ignoram a leitura da imagem e incrementam a leitura directa da tabela

Software SIG	Horas:Min:Seg:	Segundos	PTRTM[166]
ArcGIS 10.2	00:00:02,08	2,08	14,16 %
GRASS GIS 7.0.0	00:00:14,69	14,691	100 %
QGIS 2.10.1	00:00:00,691	0,691	4,70 %
GDAL 111	00:00:00,116	0,116	0,79 %

Para completar a análise falta apenas saber quanto tempo demora o ArcGIS a apresentar a solução tendo de ler a imagem como um vector (célula por célula). A resposta foi obtida com recurso à intercalação do Programa 37[167] no Programa 19[168] - substituição da def *calcula_unitArea* e da def *nr_pixel_classe* -, e é sintetizada na Figura 53.

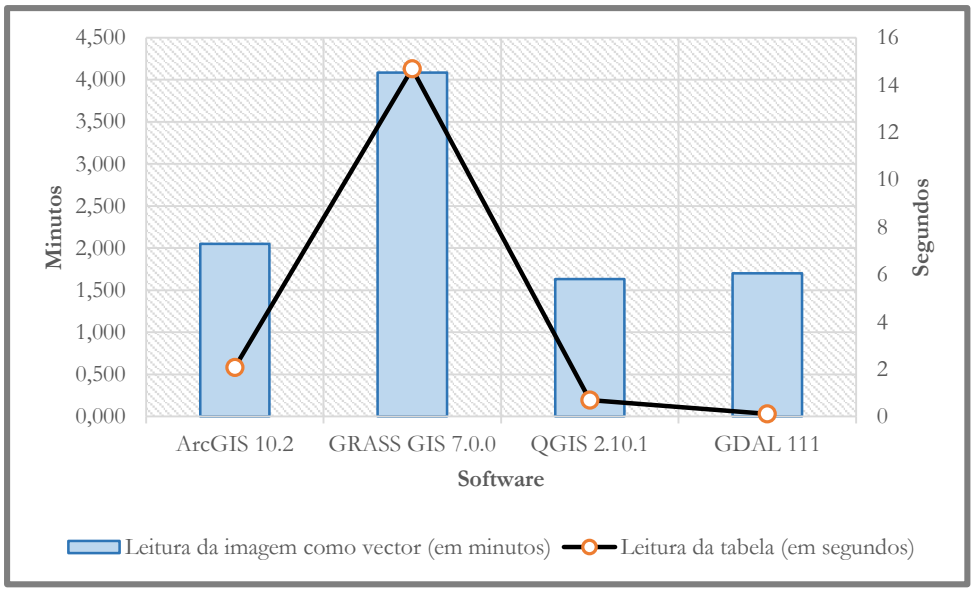


Figura 53 - Compilação e síntese dos resultados obtidos, por tipo de leitura dos ficheiros.

Em suma, demonstrámos que qualquer tipo de exercício de *benchmarking* deve considerar múltiplas abordagens, consoante as distintas propriedades de cada *software*, e não apenas a primeira que nos pareça óbvia. Depois de normalizados, os resultados mostram que o GRASS é sempre o mais lento,

[166] Percentagem de tempo despendido relativamente ao tempo máximo registado.
[167] **Ver Anexo A, p. A63.**
[168] **Ver Anexo A, p. A17.**

e que afinal a biblioteca GDAL/OGR supera, nos dois casos, o desempenho do ArcGIS, beneficiando provavelmente da minimização de recursos (do computador) utilizados, pois os seus programas correm directamente na *Shell* do editor Python utilizado. Os bons resultados do QGIS devem-se à integração da biblioteca GDAL/OGR, sem a qual não seria possível ler a imagem célula por célula.

3.2. Criação de amostras ideais e o problema da manipulação de estruturas de dados vectoriais

O problema da criação de amostras ideias é mais exigente do que o anterior, não pela complexidade do algoritmo, mas pelo considerável número de *features* que o *software* tem de processar (cfr. Tabela XXX, p.180 e XXXI, p. 182). Esta exigência está bem patente nos resultados obtidos, apresentados na Tabela XXXIV.

Tabela XXXIV - Descrição dos resultados relativos ao desempenho de cada uma das soluções na execução dos programas que intentam a criação de amostras repartidas de modo desejável[169]

Software SIG	Tempo	Descrição
ArcGIS 10.2	00:10:40	Concluiu o processo e gerou o resultado esperado.
GRASS GIS 7.0.0	01:30:00	Depois de criar todas as amostras, entrou em fase de escrutínio das mesmas; enquanto trabalhava a quarta amostra de validação (entre as seis) registou um erro e encerrou (Figura 54).
QGIS 2.10.1	04:36:00	Concluiu o processo e gerou o resultado esperado.
GDAL 1.11	03:00:00	O processo foi terminado pelo utilizador – o Programa 26 criou apenas a amostra relativa ao COS; encalhou na tarefa de divisão das várias classes do tema curvatura das vertentes em *feature classes* individuais.

Como nos elucida esta tabela, o SL/CA submetido a este teste mostrou-se incapaz de executar a tarefa proposta num período de tempo reduzido e em alguns casos não a conseguiu completar e devolver o resultado pretendido. O ArcGIS e o QGIS foram os únicos a terminá-la, pesem, no entanto, as significativas diferenças apontadas – o tempo consumido pelo primeiro diz respeito a 3,86% do período que o segundo levou a completar o exercício. O QGIS demonstra alguma ineficiência no processamento de um elevado número de *features*, à semelhança do que havíamos verificado no estudo de Patriarca *et al.* (2014a).

[169] Faz-se acompanhar esta tabela de outra informação que atesta a integridade dos Programas 23, 24, 25 e 26. Com o intuito de comprovar que estes programas funcionavam e geravam o resultado esperado, eles foram testados apenas com dois factores condicionantes (COS e litologia, os menos exigentes em termos de processamento), demorando os seguintes tempos: ArcGIS – 00:01:41; GRASS GIS – 00:01:36; QGIS – 00:03:03; GDAL – 00:01:21. Todos geraram o resultado esperado.

O GRASS GIS gerou um erro (Figura 54) e encerrou automaticamente; no entanto realizou a maior parte do procedimento (encontrava-se a quantificar o peso da quarta amostra gerada em cada classe), levando-nos a crer que este problema poderia ser contornado com a divisão do Programa 24[170] em dois, separando a tarefa de avaliação das amostras dos métodos que levam à sua criação. De qualquer dos modos, mesmo com essa divisão, o tempo que este *software* demoraria até chegar a esta fase do procedimento leva-nos a crer que não seria capaz de competir com o ArcGIS, neste caso concreto.

Figura 54 - Erro registado pelo GRASS GIS 7.0.0 enquanto executava o Programa 24 – o erro surge 1h30 depois de se ter dado início à tarefa.

Uma possível explicação para este facto pode estar na acumulação de dados em memória (memória cache) e no excessivo número de *features* a serem processadas, sendo o primeiro acentuado pelo facto de os formatos nativos do GRASS serem mais pesados, o que se traduz no total de memória ocupada pelos dados que foram sendo produzidos ao longo da execução do Programa 24 (cfr. Tabela XXXI).

Por fim, com a GDAL/OGR, pelo desconhecimento de uma ferramenta que permita individualizar cada classe numa nova *shapefile* (*split* ou *select and export*), isso só se conseguiu (com o nosso conhecimento de computação e análise espacial de que já dispúnhamos nesta fase do trabalho) com a leitura e armazenamento de todos os vértices dos polígonos em objectos Python, do tipo lista, de acordo com o seu valor (classe a que pertencem), criando posteriormente novas *shapefiles* a partir delas (nós próprios criamos a ferramenta). Neste caso, como se trata de uma leitura e posterior

[170] **Anexo A, p. A30.**

escrita vértice-a-vértice, na qual também se têm de considerar as ilhas (buracos) de cada polígono, justifica-se a maior dificuldade em concluir o procedimento - o Programa 26[171], em três horas, conseguiu apenas gerar uma amostra e quando foi parado encontrava-se ainda a processar as 143 670 *features* do tipo de polígono que resultaram da conversão da curvatura da vertente para formato vectorial, tendo em vista a individualização das várias classes em novas *layers*.

Esta questão leva-nos a destacar outro tipo de constatação que extravasa a análise simplista relativa aos tempos despendidos e fazer a ponte com o mito atribuído ao SL/CA que nos diz que é preciso ser um utilizador/programador avançado para usar este tipo de *software*. Em relação à GDAL/OGR, este mito é realidade. Não o é se utilizarmos as suas ferramentas integradas em outra distribuição, como, por exemplo, o QGIS. Não obstante, se o usarmos individualmente, por exemplo num exercício como este, temos de possuir conhecimentos avançados de programação e de computação, de modo a optimizar os procedimentos, acelerando-os e tornando-os mais eficientes.

Para além disto, são indispensáveis conhecimentos robustos sobre as estruturas de dados que viabilizam a representação de temas da superfície terrestre no computador. O seu entendimento é crucial para perceber o modo de funcionamento dos vários Programas que vão sendo apresentados. Como se sabe, existem dois tipos de estruturas de dados: o modelo de dados vectorial e o matricial (Burrough & McDonnell, 2000; Kennedy, 2009). No vectorial, o espaço é representado como uma série de entidades discretas definidas por pontos, linhas e polígonos, se bem que estes dois últimos não são mais do que um conjunto de vértices agrupados em vectores (pontos dentro de listas transpondo para o Python), tal como exemplificado na Figura 55. As suas unidades geográficas referem-se a um eixo cartesiano e são relativas a um determinado sistema de referência espacial.

Na computação de foro espacial temos de ter presente que a mesma primitiva geométrica tem especificações próprias que nos obrigam a ter certos cuidados. Na Figura 55-d estão desenhados três polígonos que podem representar entidades distintas ou, pelo contrário, simbolizar uma entidade do tipo multiparte, à qual corresponde apenas um único vector (e uma linha da tabela de atributos). A título de exemplo, o Programa 38[172] que gerámos, cria uma *feature class* a partir de coordenadas registadas num ficheiro de texto, onde consta também a identificação das entidades lineares ou poligonais e número de partes. Analisando o código verifica-se que não é ignorada a possibilidade de existirem entidades com várias linhas ou polígonos (vectores dentro de um vector).

[171] **Ver Anexo A, p. A37.**
[172] **Ver Anexo A, p. A64.**

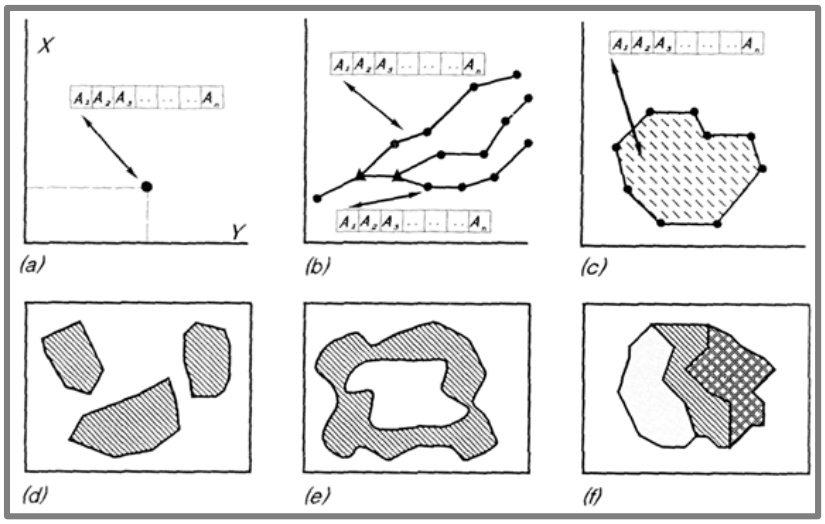


Figura 55 - Primitivas geométricas (ponto, linha e polígono) da estrutura de dados vectorial. Fonte: Burrough & McDonnell (2000), p. 22.

A elaboração deste ensaio e os vários experimentos produzidos no seu âmbito levam-nos a concluir que, tanto o QGIS como a GDAL/OGR possuem mecanismos para construir geometria multi-parte, usando uma sintaxe próxima da do Programa 38. Todavia, um algoritmo simples como este possui uma limitação. Por vezes, é frequente encontrar polígonos como o da Figura 55-e. Como se sabe, numa *shapefile*, têm de se representar todas e quaisquer fronteiras, quer seja para desenhar ou para apagar, ou seja, na criação de uma entidade geométrica como a da ilustração, ter-se-ia que desenhar os limites exteriores (no sentido do ponteiro do relógio) e apagar ou subtrair os limites interiores que formam o espaço vazio no sentido contrário ao dos ponteiros do relógio. A inclusão dos *holes* exacerba a complexidade do problema e motiva um esforço da destreza mental do operador, na medida em que nada impossibilita que, numa *feature class*, existam polígonos multi-parte com ilhas (*donuts*), o que levaria obrigatoriamente à implementação de uma estratégia de identificação daquilo que são polígonos, e espaços vazios.

Tudo o que acabamos de dizer serve dois objectivos: demonstrar a complexidade deste problema, revelando as dificuldades que se criam quando o *software* não possui uma boa ferramenta que viabilize a alteração dos dados geométricos em tarefas como as de corte topológico, agregação, ou *split* (dificuldades que acarretam para o utilizador comum questões, cujas respostas exigem conhecimentos avançados sobre estruturas de dados que ele não possui); e realçar que, naquilo que nos foi possível comprovar com este exercício e apesar das limitações evidenciadas no processamento de um grande volume de *features*, tanto a GDAL/OGR como o QGIS, à semelhança do ArcGIS, fornecem meios para manipular, na sua totalidade, a complexidade das estruturas de

dados vectoriais (no que se refere a entidades multi-parte e com ilhas). O Programa 26[173], nomeadamente a definição *criaSHPholes*, incorpora uma resolução para a gestão de temas com ilhas; na fase inicial da formulação de uma opção metodológica para a criação de amostras ideais, constatamos que o QGIS garante as mesmas possibilidades, tal como se exemplifica no Programa 39[174], instruções que fazem uma cópia de uma *shapefile*, cujas *feature* possuam ou não *holes*.

Ainda na linha dos aspectos que focam considerações qualitativas que nos parecem pertinentes, julgamos ser importante reportar alguns erros ou limitações que, pelos nossos limitados conhecimentos em computação geoespacial, não conseguimos explicar.

A ferramenta *dissolve* (agregação) não é imprescindível ou indispensável ao processo. Os nossos programas foram construídos, como já se explicou, de modo a que, a dado momento, se individualizassem os pontos que se intersectassem com cada classe. Em Python, isto implica a inventariação de todas as categorias daquele factor e a implementação de um *loop* (*for* ou *while*) para efectuar, iterativamente, a supradita intersecção e consequente divisão das entidades pontuais. Há infinitas maneiras de conseguir isso, no desenvolvimento deste exercício ocorreram-nos duas: exportar cada classe para uma nova *shapefile* e listar esses novos ficheiros, lançando um ciclo do tipo *for* (*for elemento in lista ou for elemento in range(len(lista))*); o segundo prevê a leitura da tabela da *feature class* com um cursor de procura, executando as acções de intersecção em cada iteração do ciclo. Esta última acarreta a agregação temática dos elementos geométricos em polígonos multi-parte, passando de uma situação como a apresentada na Figura 56-A para outra como a expressa na Figura 56-B, na qual, uma classe corresponde a uma linha na tabela. A segunda abordagem permite uma redução substancial do número de *features* a serem processadas.

No Programa 23[175] usou-se esta segunda abordagem, mas nos restantes incrementou-se a primeira. No GRASS GIS, a ferramenta *v.dissolve* não cria uma nova tabela vinculada ao tema agregado. Assim, para evitar o trabalho que se teria em recuperar os valores da tabela original, optou-se pela estratégia alternativa. No que diz respeito ao QGIS, a estratégia *dissolve* > *read* > *select* > *export* (idêntica à do Programa 23) foi testada e abandonada porque nos apercebemos que a ferramenta (nativa) "qgis:dissolve"[176] apresenta limitações evidentes que não somos capazes de explicar, ainda que nos atrevamos a pensar que tal se pode dever ao excessivo número de *features* existentes. Na

[173] **Ver Anexo A, p. A37.**
[174] **Ver Anexo A, p. A65.**
[175] **Ver Anexo A, p. A27.**
[176] Uma vez que o QGIS integra no seu núcleo diferentes distribuições, ao nomear qualquer uma das ferramentas que consta da sua biblioteca de algoritmos, utiliza-se o mesmo método de identificação do módulo *processing* deste *software*, de maneira a que não hajam dúvidas sobre qual o algoritmo a que nos referimos, sabendo-se se se trata de uma ferramenta nativa do QGIS, ou não.

primeira versão do Programa 25[177], este *software* criou apenas a amostra relativa ao COS, em 03:00:00, bloqueando na tarefa de agregação do tema vectorial que representa a curvatura das vertentes, mostrando-se incapaz de processar num período de tempo admissível, as 143 670 *features* que constituem o tema.

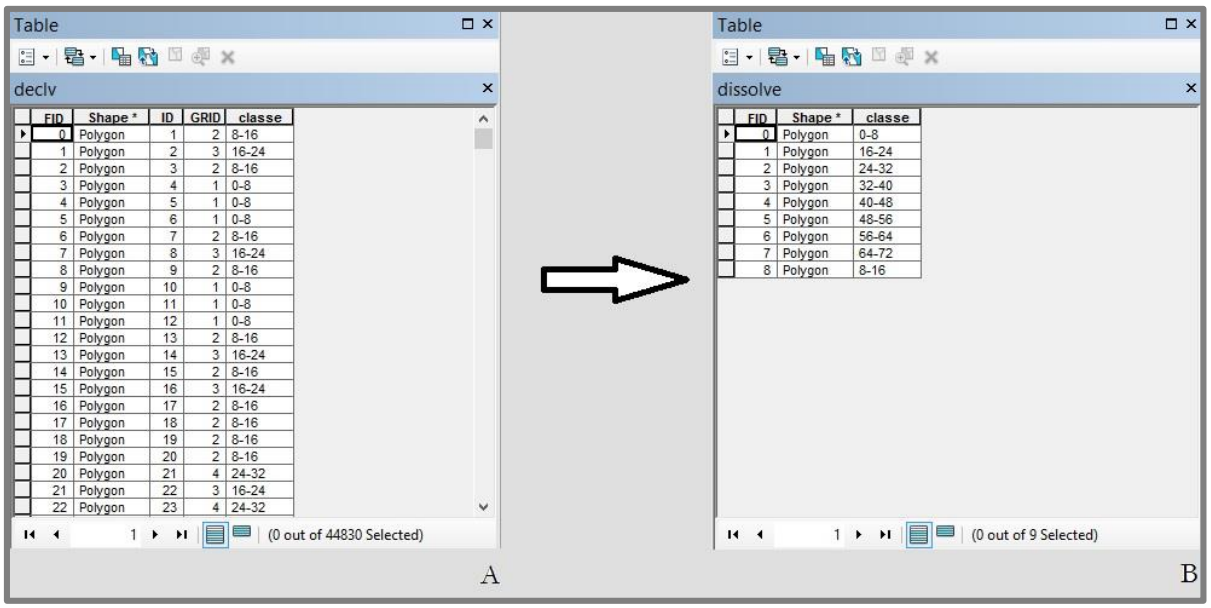


Figura 56 - Captura de ecrã relativa ao processo de agregação de um tema vectorial, tendo em vista a compactação da sua tabela, permitindo a iteração dos seus valores sem repetição dos mesmos.

Por sua vez, reconhecemos que o Programa 26 não é a opção mais eficaz para resolver o problema proposto com GDAL/OGR. Referimo-nos já sobre o desconhecimento, no momento de construção do supramencionado bloco de código, sobre a ferramenta *ogr2ogr*, que viabiliza a agregação de polígonos, ou a sua selecção e exportação. Embora a sua utilização não seja possível em Python (funcionalidade reservada à integração noutros pacotes e à utilização por via da linha de comandos do sistema operativo), a adaptação da solução equivalente proposta pelos desenvolvedores do projecto (https://pcjericks.github.io/py-gdalogr-cookbook/vector_layers.html#filter-and-select-input-shapefile-to-new-output-shapefile-like-ogr2ogr-cli, acedido em 19 Outubro, 2015), que recorre ao método `SetAttributeFilter()` da classe `OGRLayer`, talvez fosse mais eficaz na resolução deste problema, no caso de haver uma optimização da leitura das várias *features*. Ou então, por não precisar de ler todos os vértices dos polígonos.

[177] **Ver Anexo A, p. A34.**

Deixando as questões relacionadas com a agregação de polígonos, reportamos agora outros erros. No Programa 25 usámos a ferramenta nativa "qgis:splitvectorlayer", que apresenta três parâmetros a definir pelo utilizador: o tema de entrada, o campo segundo o qual esse tema vai ser partido e o *workspace* onde serão guardadas os novos temas, que são em número igual ao total de valores únicos no supracitado campo. Um olhar atento para a gramática deste bloco de código fará com que se perceba que se definiu como *workspace* de saída o local do computador onde se encontrava o *input*, isto porque independentemente do *workspace* de saída que fosse definido, o QGIS insistentemente colocava os *outputs* na pasta onde se encontrava a entrada. Mas, o mais curioso é que ao correr a def *split* isoladamente, descontextualizada do objectivo final do Programa 26, o *workspace* de saída, seja ele qual for, era assumido e o resultado era o esperado.

E não é só este algoritmo a evidenciar problemas, o "qgis:intersection" também aparenta ter sintomas de mau funcionamento (funcionamento condicionado). A Figura 57-A apresenta um polígono multi-parte; ao invés, a Figura 57-B ilustra o mesmo polígono repartido em duas *feature classes* - polígonos *single part*. Se executarmos a *tool* "qgis:intersection", usando como camada de entrada o conjunto de ocorrências de movimentos de vertente (distribuídos por uma área mais extensa que a dos polígonos ilustrados) e como camada de intersecção o tema da Figura 57-A, não obteremos qualquer resultado; no entanto, se substituirmos a *layer* de intersecção por qualquer uma das *feature class* da Figura 57-B, obter-se-á o resultado esperado.

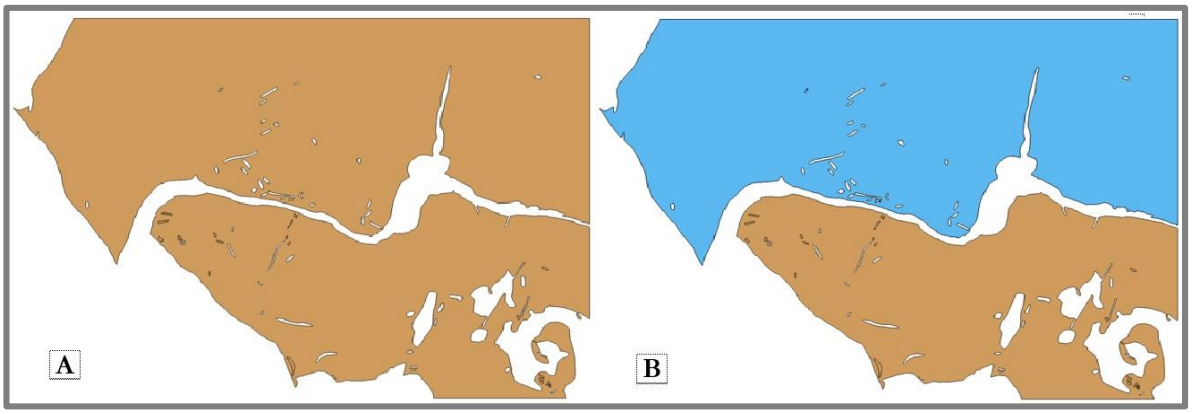


Figura 57 - Dois polígonos que representam a mesma categoria enquanto entidade geométrica individual - multi-parte (A); dois polígonos que representam a mesma categoria enquanto entidades geométricas independentes (B).

3.3. Produção de cartografia de susceptibilidade a movimentos de vertente através da incrementação do método do valor informativo

Tendo como ponto de partida a amostra de treino derivada do processo anterior, estávamos em condições de executar os blocos de código que incrementam o método do VI.

À semelhança do exercício anterior, registaram-se problemas que inviabilizaram a conclusão do exercício em dois *software*. Segundo a Tabela XXXV, o ArcGIS foi mais uma vez o *software* mais rápido, demarcando-se claramente do GRASS, demorando 8,74% do tempo total gasto por este *software*. A nível de processamento, reconhecemos que este exercício é menos exigente que o anterior, o que justifica certamente um menor esforço sob a memória cache, permitindo ao GRASS concluir a tarefa, e mais do que isso, destacar-se relativamente às outras duas alternativas. No ensaio anterior, este *software* apresentou um erro, mas, se não o tivesse feito, em menos de duas horas concluiria o processo, ainda que tal desempenho represente metade do do QGIS, e dois terços daquilo que do da GDAL/OGR, apesar de não ter chegado a metade do procedimento.

Tabela XXXV - Descrição dos resultados relativos ao desempenho de cada uma das soluções na execução dos programas que intentam a produção de um cartograma que representa a susceptibilidade do território à ocorrência de movimentos de vertente

Software SIG	Tempo	Descrição
ArcGIS 10.2	00:03:32	Concluiu o processo e gerou o resultado esperado.
GRASS GIS 7.0.0	00:40:25	Concluiu o processo e gerou o resultado esperado.
QGIS 2.10.1	Erro ao executar	Não foi possível executar o processo porque a ferramenta "gdalogr:rasterize" não funciona correctamente no sistema operativo Windows.
GDAL 1.11	03:00:00	O processo foi terminado pelo utilizador – o Programa 30 ficou preso na intersecção entre os movimentos e as classes do tema curvatura das vertentes.

As diferenças entre o GRASS e o ArcGIS resultam, provavelmente, não da qualidade dos algoritmos geoespaciais envolvidos ou da sua capacidade de processamento, mas sim do peso em disco e na memória dos dados convertidos e gerados no formato nativo do GRASS, claramente mais pesados que imagens *.tif* ou vectores *.shp*. Recordamos a Tabela XXXI, na qual é visível a significativa diferença de tamanho dos *outputs* intermédios gerados pelos Programas 27[178] e 28[179] – em que o ArcGIS processou apenas 20,13% do volume de dados que o GRASS geriu.

Em suma, parece-nos que o GRASS é um SIG *desktop* com mais capacidades quando o objectivo é a modelação espacial, pois dispõe de uma biblioteca de algoritmos geoespaciais mais rica e é mais rápido que outras alternativas livres, pese embora o facto de permanecer inferior ao *software* da

[178] **Ver Anexo A, p. A43.**
[179] **Ver Anexo A, p. A45.**

ESRI, sobretudo porque os seus formatos nativos são demasiado pesados, algo que sobrecarrega facilmente a memória cache e que deve estar na origem, por vezes, de erros inesperados.

As Figuras 58 e 59 correspondem ao *output* final dos Programas 27 (ArcGIS) e 28 (GRASS GIS), respectivamente. A interpretação visual e a análise estatística das imagens permitem detectar uma distribuição de valores idêntica, comprovada por um coeficiente de correlação linear de valor "1" (calculado pelo Programa 11[180]), isto é, não há quaisquer distorções motivadas por funcionamentos incorrectos de ferramentas análogas. No entanto, subtraindo as duas imagens, percebem-se pequeníssimas diferenças (a partir da sexta casa decimal) entre os dois cartogramas, justificadas por algumas diferenças no código que as originou, mas que não têm qualquer impacto na integridade lógica dos *outputs*.

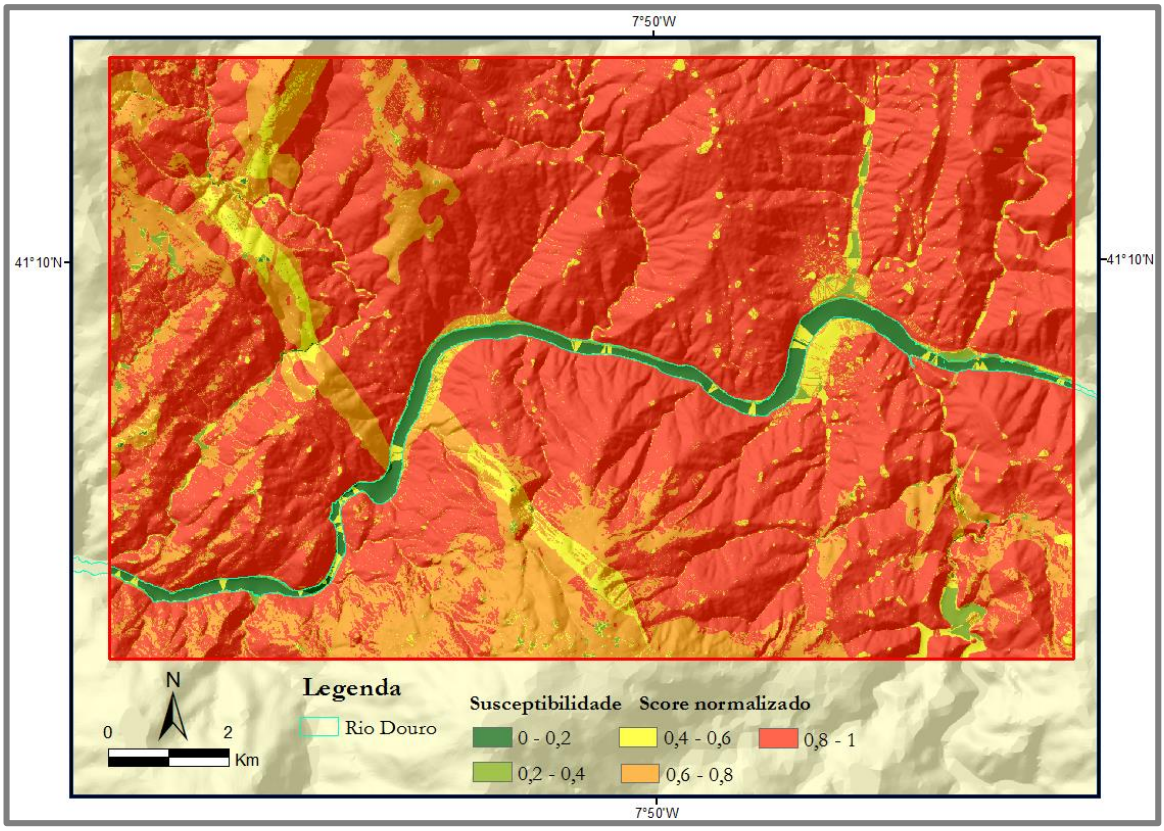


Figura 58 - Cartograma de susceptibilidade à ocorrência de movimentos de vertente na área de estudo - resultado do Programa 27 – ArcGIS 10.2.

No Programa 27, os VI de cada classe foram registados na tabela anexa à imagem de cada factor condicionante, portanto, a responsabilidade pelos arredondamentos é do próprio *software*. Por seu

[180] **Ver Anexo A, p. A10.**

turno, no Programa 28, os VI foram guardados num dicionário com o valor de uma chave única (identificação da categoria), para além disto, como a ferramenta de reclassificação do GRASS exige que os novos valores sejam inteiros, multiplicaram-se os VI do dicionário por 1 000 000. Para comparar os dois cartogramas, tivemos de dividir o *raster* originado pelo GRASS, pelo mesmo valor. Tudo isto fundamenta as distorções entre as duas imagens, ínfimas na medida em que resultam de problemas de arredondamento.

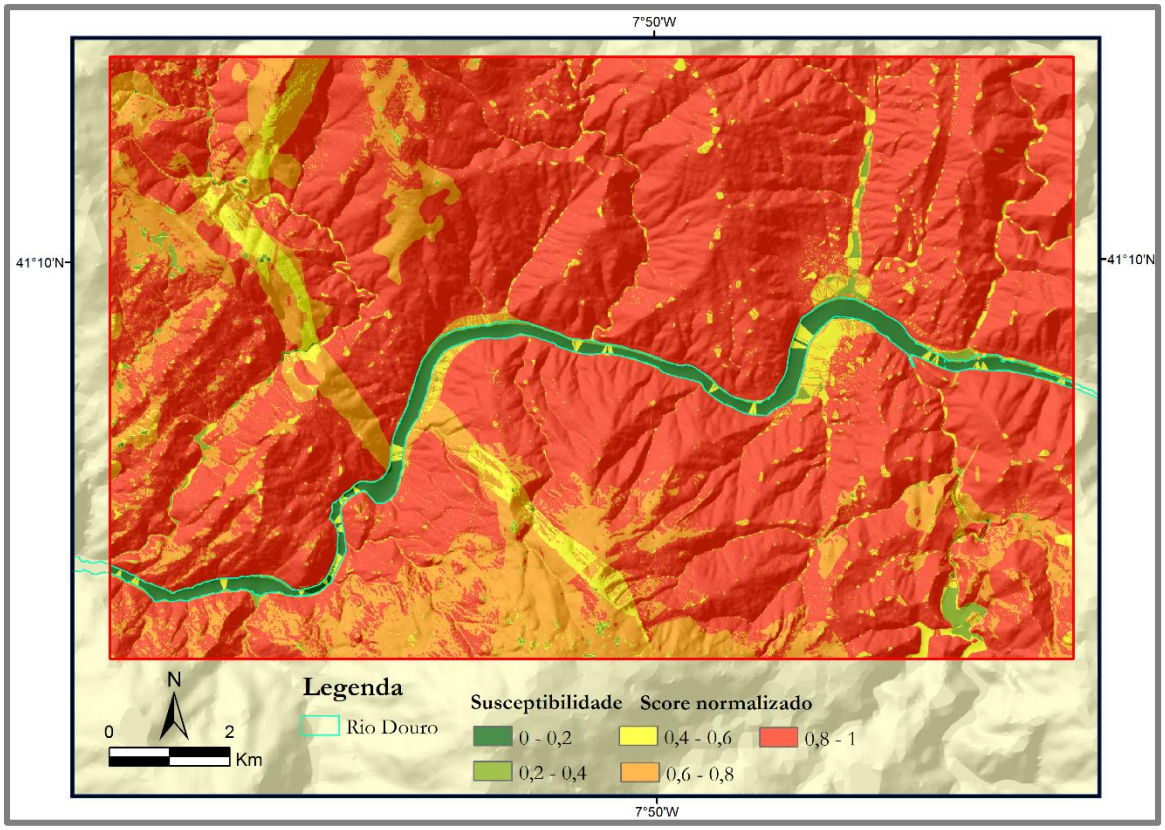


Figura 59 - Cartograma de susceptibilidade à ocorrência de movimentos de vertente na área de estudo - resultado do Programa 28 – GRASS GIS 7.0.0.

No que diz respeito ao QGIS, constatou-se que a ferramenta "gdalogr:rasterize" integrada no primeiro não corre correctamente no sistema operativo Windows 8.1. O programa tenta executar a ferramenta, a API não devolve qualquer erro, mas o *output* não é gerado. Ao testar-se individualmente a ferramenta, percebemos que em alguns momentos ela funciona normalmente, noutros não devolve o *output* esperado, sem qualquer tipo de razão aparente. Este erro quebra a cadeia de processos e não permite a conclusão do procedimento. Curioso foi constatarmos que,

adaptando o Programa 29[181], e executando-o no Ubuntu, o procedimento corre dentro da normalidade.

A nossa experiência com a utilização do QGIS demonstra que este funciona melhor em Linux, isto é, muitas das ferramentas que evidenciam limitações no Windows, não apresentam quaisquer problemas quando usadas, por exemplo, em Ubuntu. Este é mais um exemplo que se encontra em conformidade com o relatado no parágrafo anterior.

A juntar a isto, não podemos esquecer a dependência que o QGIS evidencia relativamente a outras distribuições. Por exemplo, não existe um algoritmo nativo para reclassificação de um *raster;* existe uma calculadora *raster* nativa, contudo não encontramos forma de a invocar com a API Python, o que nos obrigou a utilizar a calculadora do GRASS, que no QGIS está limitada em número de *inputs* (dá-nos a opção de escolher até seis imagens matriciais). Enfim, os desenvolvedores do QGIS levam a interoperabilidade muito a sério, tanto que fazem deste *software* um simples visualizador e um repositório de ferramentas de outras distribuições, um esforço difícil de compreender quando o GRASS GIS já ostenta uma biblioteca de algoritmos tão diversificada e completa.

Por fim, no que concerne à GDAL/OGR, importa, em primeiro lugar, ter presente que a construção dos vários procedimentos utilizados resultou de uma metodologia de trabalho autónoma e isolada, que implicou um grande espírito de autodidactismo (no qual o apoio da comunidade teve um contributo diminuto e pontual), e que partiu de uma base de conhecimentos relativos ao SL/CA SIG praticamente inexistentes ou básicos. Por conseguinte, a aprendizagem foi conseguida através de um processo de Thorndikeano[182] de tentativa-erro-aprendizagem, que nos levava a analisar criticamente os resultados e a procurar outras abordagens, documentadas ou não pela comunidade. O Programa 26[183], relativamente ao qual reconhecemos a sua incapacidade de ser eficiente, apontando outra forma de aportar o problema, é um exemplo disso mesmo. Dadas as semelhanças entre o algoritmo de criação das amostras ideais e o da aplicação do método do VI, foi possível, neste exercício, experimentar, no Programa 30[184], a supramencionada alternativa metodológica para a desagregação das várias categorias de um tema em ficheiros individuais (transporte da ferramenta *ogr2ogr* para Python).

[181] **Ver Anexo A, p. A49.**
[182] Edward Lee Thorndile foi um psicólogo Norte-americano que esteve na origem do surgimento do condicionamento operante, isto é, descobriu que um ser vivo em resposta a uma consequência agradável tende a repetir o comportamento e faz exactamente o contrário quando recebe uma consequência desagradável.
[183] **Ver Anexo A, p. A37.**
[184] **Ver Anexo A, p. A51.**

Concluiu-se que esta técnica supera a limitação do Programa 26, já que deste modo, a *shapefile* da curvatura das vertentes, sendo cisionada em vários ficheiros de um modo célere, permitiu resolver o problema. No entanto, este procedimento motivou um novo condicionalismo no método que realiza a intersecção entre dois temas poligonais. Na execução do Programa 30, a GDAL/OGR ficou retida na intersecção entre os movimentos e uma das classes do tema curvatura das vertentes, em 3 horas, revelando mais um *handicap* do Programa 26, que não seria executado mesmo com a resolução do problema do *split* do tema vectorial mencionado.

Mais uma vez, não se pode deixar de exaltar a dificuldade que envolveu a construção do Programa 30. A inexistência de uma calculadora *raster* exige ao operador capacidades avançadas de programação para a estipulação de definições que implementem o conceito da calculadora *raster*. Ainda assim, a concepção de uma calculadora matricial encontra-se facilitada pela boa documentação existente que, depois de conhecer a estrutura da imagem vista enquanto um conjunto de vectores/listas, nos permitiu iterar as diferentes matrizes, somando os valores que partilham a mesma posição, para depois se criar uma nova imagem a partir de um vector (*numpy array*) - https://pcjericks.github.io/py-gdalogr-cookbook/raster_layers.html#create-raster-from-array (acedido em 8 Novembro, 2015).

Esta ferramenta (*create raster from array*), adaptada à soma de seis imagens matriciais, apresenta um desempenho bastante aceitável. Com ela conseguimos realizar operações aritméticas entre os vários pixéis das seis imagens em cerca de 20 segundos. Isto demonstra que a biblioteca GDAL/OGR é mais eficiente em processamento de imagens matriciais do que no processamento de objectos geométricos vectoriais.

3.4. Avaliação da exactidão global do modelo, síntese dos resultados e outros apontamentos sobre a versatilidade do SL/CA

Entramos na última fase do processo de modelação da susceptibilidade na área delimitada pela folha n.º 126 da Carta Militar do Exército (1/25 000). Nenhum exercício deste tipo estaria completo sem o incremento de uma qualquer estratégia de validação do modelo. Neste caso, tal como já foi dito, usaremos a análise ROC e a AUC como medida global de exactidão, técnica que se desdobra em dois momentos:

i. Execução de uma cadeia lógica de instrumentos que visa a adição das ausências à amostra de validação criada e escolhida anteriormente, associando a cada um dos

pontos os valores coincidentes que constam no cartograma de susceptibilidade (Programas 31[185], 32[186], 33[187] e 34[188]);

ii. Construção do gráfico ROC com recurso a *software* especializado em estatística.

O primeiro momento diz respeito a uma cadeia de processos relativamente simples e pouco exigente que, mais do que avaliar a performance, avalia a inexistência de ferramentas (como a diferença entre polígonos, extracção de valores para pontos e criação de pontos aleatórios, etc.) nas distribuições *open source* em confronto, bem como outros aspectos relacionados com a orgânica dos vários programas construídos. Ainda assim, no gráfico da Figura 60, não deixamos de apresentar os tempos gastos por cada *software*.

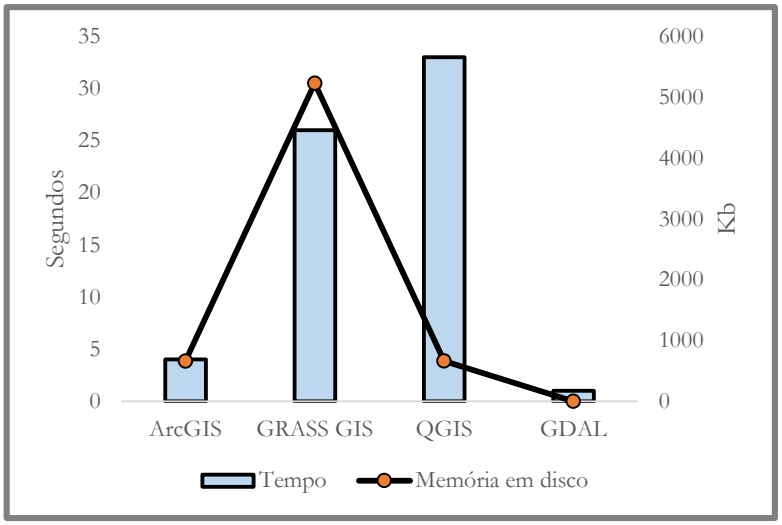


Figura 60 - Tempos gastos por cada *software* na preparação da tabela com os dados necessários para a avaliação da exactidão global do modelo.

A biblioteca GDAL/OGR foi a mais rápida (1 segundo), seguindo-se o ArcGIS (4 segundos), o GRASS (26 segundos) e, por último, o QGIS (33 segundos). Estes resultados merecem-nos duas considerações relevantes; a primeira leva-nos a reflectir sobre as diferentes estratégias usadas para se obter idêntico resultado; a segunda, prende-se com a morosidade latente de algumas distribuições.

Até aqui, as várias rotinas apresentadas têm os mesmos objectivos mas gramáticas diferentes. Em diversas situações, devido à inexistência ou mau funcionamento de um algoritmo de

[185] **Ver Anexo A, p. A56.**
[186] **Ver Anexo A, p. A57.**
[187] **Ver Anexo A, p. A59.**
[188] **Ver Anexo A, p. A61.**

geoprocessamento, fomos forçados a procurar outras abordagens para atingir a mesma meta/objectivo. Trabalhar com SL/CA é exactamente isto, adaptabilidade e interoperabilidade, gerando e aproveitando sinergias processuais e metodológicas.

Por exemplo, a inexistência, no GRASS GIS (Programa 32[189]), de uma ferramenta que devolve a diferença entre dois polígonos (análoga ao *erase* do ArcGIS 10.2) levou-nos a incrementar uma estratégia que contorna esse *handicap*, propondo uma resolução que passou pela utilização do formato matricial para remover as áreas limítrofes às ocorrências da superfície sobre a qual se pretendem criar as ausências como pontos aleatórios. Isto, por si só, implicou a execução de um maior número de ferramentas (união, conversão para formato matricial, reclassificação, etc.) – desnecessárias nos restantes programas - e o processamento de um maior número de células. Mas, como se tal não bastasse, encontrámos outro problema; o GRASS possui duas ferramentas que permitem criar pontos aleatórios, a *r.random* e a *v.random*, sendo que a primeira consente a criação destes pontos sobre uma superfície matricial. Uma vez que estamos a trabalhar com *raster*, preferimos a *r.random*, no entanto, esta não funcionou como era previsto, inviabilizando a concretização de toda a cadeia (o *software* bloqueia e tem de ser encerrado). Assim, necessitamos de uma nova conversão para formato vectorial, de modo a conseguirmos usar a *v.random* que, por seu turno, não apresentou qualquer problema de funcionamento.

Tudo isto justifica as diferenças do Programa 32, relativamente aos demais, a nível do número de ferramentas utilizadas, sobretudo no que diz respeito ao número de linhas, número de células processadas, e complexidade.

Quanto à GDAL/OGR (Programa 34[190]), mais uma vez se comprovou a dificuldade em operar esta alternativa tendo em vista o processamento encadeado de dados geográficos. A inexistência de todas as ferramentas necessárias, e usadas nos outros programas, levou-nos a conceber vários métodos para a criação de pontos aleatórios não coincidentes com as ausências, recorrendo-se ao módulo *random* (do Python) combinado com as informações sobre a extensão dos limites da área de estudo (distância à meridiana e à perpendicular, mínimas e máximas). Este procedimento garante maior eficiência, pois não implica a utilização de ferramentas como as executadas nas demais rotinas, bem como não cria qualquer *output* intermédio, ou melhor, todos os dados intermédios são listas e outros objectos do *Python*. Porém, exige conhecimentos avançados de programação e tempo, que, por sua vez, é mal gasto, uma vez que as diferenças de tempo não justificam o esforço de construção da rotina.

[189] **Ver Anexo A, p. A57.**
[190] **Ver Anexo A, p. A61.**

Em segundo lugar, e sintetizando o que se disse até agora sobre a morosidade das várias distribuições, parece-nos claro que o QGIS é muito pouco eficiente em processamento de dados geográficos, algo que se verifica tanto neste como no exercício das amostras ideais. Se se tratar de um volume reduzido de dados, as diferenças para o SP são menos perceptíveis; no entanto, elas tornam-se gritantes com o aumento do volume de dados.

Por sua vez, combinando as informações dos vários exercícios, concluímos que a biblioteca GDAL/OGR é uma solução muito forte quando se trata de processamento de um reduzido volume de dados, sendo incapaz de processar em tempo útil dados vectoriais com dezenas de milhar de *features*. Julgamos também que o seu desempenho é mais positivo quando se trabalha com dados matriciais.

Entrando na segunda parte deste exercício, analisámos os grafismos produzidos no âmbito da análise ROC em dois *software* de estatística (SPSS e R), e que constam da Figura 61. Nota-se que são idênticos, contudo os procedimentos que lhe dão origem são completamente diferentes em dificuldade e "amigabilidade". No SPSS, como vimos anteriormente, basta fazer executar uma única ferramenta (cfr. Figura 51, p. 172).

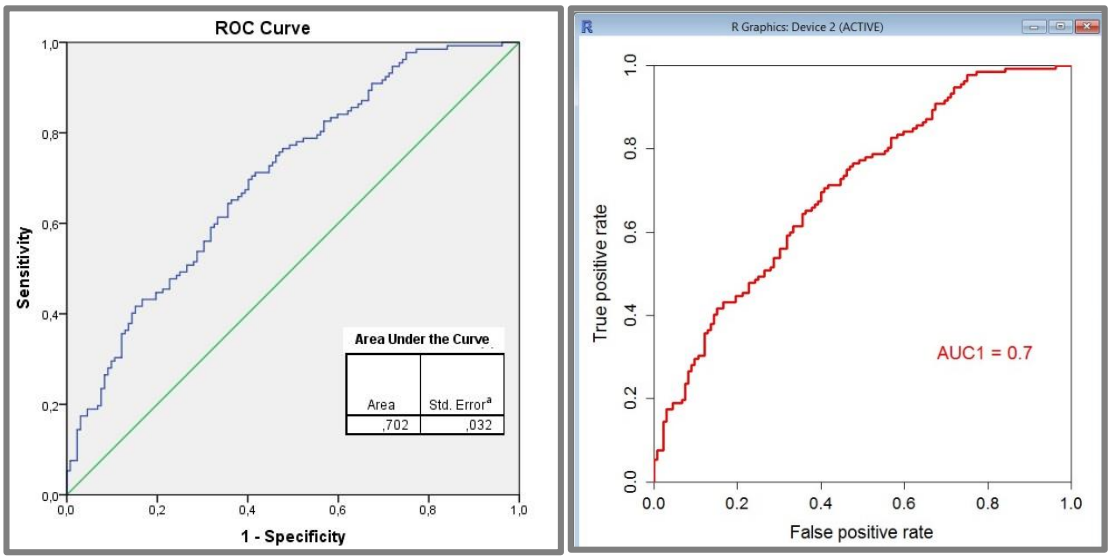


Figura 61 - Gráfico ROC e medida AUC – SPSS, à esquerda e R *Statistics,* à direita.

No R, o operador tem trabalho acrescido porque, em primeiro lugar, tem de actualizar alguns pacotes do *software*, instalando o ROCR para, posteriormente, executar um *script* que constrói a curva ROC e devolve o valor da AUC. Se este conjunto de instruções for disponibilizado pela

comunidade, o trabalho é facilitado, mas se tal não suceder, o operador tem de desenvolver ou aplicar um conjunto de competências que lhe permitam construir uma rotina deste género.

Para além disto, o operador encontrará adversidades, muitas vezes, de difícil explicação e, por isso, difíceis de contornar. Por exemplo, segundo a nossa experiência - este *script* (Programa 18[191]), em algumas máquinas não funciona, situação para a qual nunca fomos capazes de encontrar explicação, uma vez que o sistema operativo, a versão e o modo de instalação do R eram exactamente os mesmos.

Ainda assim, embora existam maus exemplos (a ferramenta ROC Curve do Sextante não funciona correctamente, como se pode ver na Figura 62), existem outros SL/CA SIG que disponibilizam ferramentas deste tipo, só temos de olhar para todas elas numa lógica de interoperabilidade.

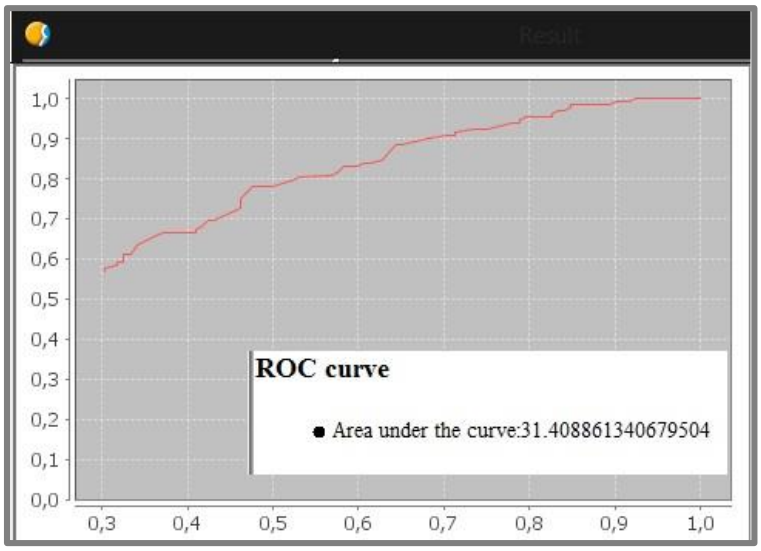


Figura 62 - Gráfico ROC e medida AUC gerada pelo algoritmo ROC *Curve* da *toolbox* Sextante incorporada no gvSIG.

Em síntese, independentemente dos resultados apresentados neste exercício, o que daqui resulta é a conclusão de que o SL/CA é uma solução capaz de resolver os problemas quando considerada numa lógica de interoperabilidade. Nestes testes seguimos condições e regras rígidas, nas quais inviabilizávamos e desvalorizamos a cooperação entre ferramentas de distribuições distintas, visto que o nosso propósito era o confronto entre soluções, o que exigia o uso de ferramentas nativas e a estipulação de ensaios completos, sem segmentação dos procedimentos em programas mais pequenos, e sem interoperabilidade.

[191] **Ver Anexo A, p. A16.**

Todavia, não nos podemos esquecer de que no mundo real estas regras não se aplicam, visto que somos livres de combinar funcionalidades e fazer convergir sinergias. Assim, e perante os resultados obtidos, enfatizamos não a conotação do SL/CA como uma plataforma de fraca qualidade, mas, antes, o facto de permitir a resolução de problemas concretos se usadas de modo interoperável. No entanto, alertamos para algumas dificuldades, porque não é propriamente fácil, pois a interoperabilidade exige um operador versátil, capaz de manipular várias distribuições simultaneamente, canalizando o melhor de cada uma para a resolução do problema, algo que, sem dúvida, exige conhecimentos avançados, não só para executar mas sobretudo, para interpretar os resultados (e os sinais a eles associados) que a tecnologia lhe dá, indispensáveis para que lhe seja possível perceber e corrigir o que pode estar a correr mal.

Apesar de, em alguns momentos, o SL/CA poder ser de difícil manuseamento, é uma solução capaz e fiável, embora nem sempre performante como se desejaria, facto que deve ser encarado pelos desenvolvedores e pela comunidade como uma realidade que necessita de urgente alteração.

CAPÍTULO III – EXERCÍCIO 2: MODELAÇÃO DA ACESSIBILIDADE GEOGRÁFICA RELATIVA A UNIDADES DE SAÚDE EM SITUAÇÕES DE EMERGÊNCIA MÉDICA

A legislação portuguesa é peremptória ao defender que todos os cidadãos têm o direito à protecção de um serviço de saúde universal tendencialmente gratuito. Assim sendo, a organização da rede de infra-estruturas de saúde e dos seus meios de resposta às necessidades da população é de grande relevância para o Ordenamento do Território, ainda mais quando, em algumas partes do território continental português, muitos habitantes se encontram isolados relativamente a serviços de saúde especializados. A elevada complexidade topográfica do espaço euclidiano e a desigual distribuição das unidades de saúde fazem com que os indivíduos revelem diferentes condições de acesso (geográfico) a estes bens fundamentais, de tal modo que esta questão tem sido uma preocupação crescente de diferentes governos, países e sociedades, com expressão em vários trabalhos onde se apresentam processos de medição de acessibilidade geográfica a este tipo de equipamentos/serviços, recorrendo-se a índices sintéticos, de maior ou menor complexidade e incorporando um conjunto amplo de variáveis medidas com maior ou menor objectividade e exactidão (Joseph & Phillips, 1984; Martin *et al.*, 2002; Luo & Wang, 2003; Rosero-Bixby, 2004; Tanjer *et al.*, 2006; McGrail & Humphreys, 2009 e 2014; Ngui & Vanasse, 2012; Yao *et al.*, 2013; Kanuganti *et al.*, 2014; Aoun *et al.*, 2015).

Em Portugal, estes problemas integram-se, por exemplo, nas competências de órgãos como a Direcção Geral da Saúde, o INSA ou o INEM (Decreto Regulamentar 14/2012, Regulamento 329/2013, Decreto-Lei 34/2012, respectivamente).

Neste sentido, imagine-se um cenário em que a Unidade de Promoção da Saúde do Departamento de Promoção da Saúde e Prevenção de Doenças Não Transmissíveis do INSA, em parceria com o INEM, procura averiguar as desigualdades existentes no acesso a serviços e infra-estruturas de saúde e na recepção de apoio médico em situações de emergência, tendo como principal objectivo a elaboração de um plano de acção que colmate quaisquer dificuldades de acesso e dirija esforços para a sensibilização das populações mais remotas para certos cuidados de saúde que deverão ter, e identificando também as populações mais vulneráveis e mais distantes das equipas de socorro, reflectindo sobre medidas que possam minimizar essa distância.

Em termos formais, o problema pode ser debatido numa estrutura tri-vectorial com respectivo cortejo de objectivos/propósitos e tarefas, tal como seguidamente se apresenta:

1. **Modelação da acessibilidade física aplicada à geografia da saúde: o conceito de acessibilidade, estruturas de dados e resenha dos procedimentos e tarefas processuais**
   i. Esclarecer, de modo sucinto e breve, o conceito de acessibilidade física ou geográfica.
   ii. Contrapor as duas estratégias comummente utilizadas no cálculo de um indicador de acessibilidade física.
   iii. Delimitar as áreas de teste.
   iv. Estabelecer uma linha processual e operacional que identifique as várias etapas do exercício de produção de cartografia de acessibilidade a infra-estruturas de saúde, tendo em conta a capacidade de resposta das equipas de emergência médica em situações de emergência.
   v. Justificar e explicar cada etapa processual, num contexto em que se tentam analisar os comportamentos do *software* com o aumento gradual do volume de dados.
2. **Base analítica para selecção de *software* com funcionalidades avançadas para manipulação de dados matriciais para implementação do modelo de cálculo da acessibilidade em ambiente SIG**
   i. Organizar um conjunto de critérios que respondam à necessidade de selecção das alternativas a testar.
   ii. Valorar e ponderar os vários atributos e objectivos.
   iii. Apresentar os *software* seleccionados.
   iv. Caracterizar os programas construídos com o fim de produzir cartografia de susceptibilidade nos vários *software* seleccionados.
   v. Identificar as principais propriedades dos dados de entrada.
3. **Comparação de desempenho entre *Software* SIG Proprietário e *Software* SIG Livre: 2ª Análise**
   i. Apresentar e discutir os resultados de performance obtidos pela execução dos vários Programas.

ii. Apontar possíveis limitações, erros ou falhas que se venham a verificar.

iii. Relacionar os resultados com as diferenças dos vários programas.

I. Modelação da acessibilidade física aplicada à geografia da saúde: conceito de acessibilidade, estruturas de dados e resenha dos procedimentos e tarefas processuais

"A acessibilidade surge como elemento fundamental para garantir os princípios de equidade territorial""

Rui Pedro Julião (2001:208)[192]

De acordo com Ferreira (2012), o conceito de acessibilidade apresenta uma simplicidade enganadora: por um lado, há uma grande diversidade de critérios que podem ser utilizados para a calcular; por outro, o conceito, em si mesmo, pode ser entendido numa multiplicidade de perspectivas e aplicado de forma diferente em infinitos contextos. No presente exercício consideramos apenas a acessibilidade física ou geográfica, entendida enquanto capacidade/esforço de se alcançar determinado lugar a partir de qualquer outro ponto e independentemente do meio transporte. Este esforço/distância possibilita uma classificação morfológica do espaço geográfico, medida em unidades de comprimento e tempo (custo ou impedância da rede). Em geografia da saúde, esta perspectiva sobre o conceito de acessibilidade define aquilo que se chama de acessibilidade potencial, na medida em que valoriza sobretudo as impedâncias e obstáculos físicos – em oposição, a acessibilidade revelada é a acessibilidade real e engloba outras variáveis sociodemográficas[193].

Em termos metodológicos, na modelação da acessibilidade física ou geográfica existem, essencialmente, dois modos de abordagem mais comuns:

[192] JULIÃO, Rui (1994) – *Tecnologias de Informação Geográfica e Ciência Regional – Contributos Metodológicos para a Definição de Modelos de Apoio à Decisão em Desenvolvimento Regional.* **Dissertação de Doutoramento**, Faculdade de Ciências Sociais e Humanas da Universidade Nova de Lisboa, 328 p.

[193] A acessibilidade revelada é aquela que realmente se verifica, uma vez que o comportamento perante os serviços de saúde (processo através do qual cada pessoa que percebeu em si próprio estar doente escolhe e implementa estratégias para facilitar a recuperação) não depende só da distância, sendo uma medida global que integra a distância física com variáveis sociais, demográficas e culturais. Assim, é limitativo considerar apenas a distância entre o prestador e consumidor, não só pelo que se disse, mas também porque proximidade não garante elevada acessibilidade porque esta depende do número de pessoas que compete pelos serviços disponíveis (McGrail & Humphreys, 2009 e 2014).

i. Os que se baseiam na teoria dos grafos[194], que remonta ao problema das 7 Pontes de Konigsberg proposto por Euler (Rocha, 2012) – modelo matemático no qual se considera uma rede de eixos (vectores) que interconectam nós, sobre a qual são definidos os parâmetros de movimentação, as impedâncias e os obstáculos absolutos;

ii. E, os que recorrem ao modelo matricial de dados, que se baseia na segmentação do espaço numa malha regular de entidades contíguas, e, no qual, todos os pontos do espaço, ao contrário do que acontecia no primeiro, ficam abrangidos pelo modelo, sendo os parâmetros de movimentação, as impedâncias e os obstáculos incorporados em cada unidade matricial (Ferreira, 2012).

Ambas as alternativas têm vantagens e desvantagens. Com a utilização do modelo vectorial, os valores de acessibilidade devolvidos ficariam confinados a um conjunto de eixos e de nós enquanto que, ao invés, no modelo matricial cada uma das células dentro da área de estudo apresenta um valor de impedância. Para além disto, neste tipo de modelo de dados é possível a integração de outras variáveis – distribuição espacial de outros factores com peso na deslocação, como por exemplo os declives ou o uso e ocupação do solo (Julião, 2001).

Porém, se existirem vários níveis de fluxo que se cruzam ou sobrepõem no espaço segundo interconexões que obedecem a regras específicas, está-se perante um problema de difícil resolução (Ferreira, 2012). Por exemplo, no plano é frequente a existência de cruzamentos entre as vias primárias (auto-estradas e outras vias de alta velocidade) e secundárias (regionais e nacionais) que, na realidade, não constituem uma conexão real entre o fluxo de cada uma delas, ou seja, o agente móvel não pode entrar ou sair da via primária por esse ponto.

Numa situação como esta em que trabalhamos com um grafo não planar, a utilização de vectores seria recomendável, visto que existem vários *software* que tornam possível a modelação da elevação das vias. Por outro lado, o modelo matricial também anula a possibilidade de integração de regras de trânsito, sejam elas sentidos proibidos ou proibições de inversão do sentido de marcha.

Ainda assim, neste ensaio, optou-se pelo emprego do modelo matricial, por quatro motivos essenciais:

[194] Na análise de redes, os procedimentos envolvem apenas (ou sobretudo) as relações de conectividade representadas no espaço relativo onde não interessa a localização absoluta das infra-estruturas, mas sim os percursos e fluxos (espaço leibnitziano). Um dos atractivos do modelo de redes é que o seu suporte (ou origem) matemático (teoria dos grafos) constitui uma área de pesquisa consolidada. Um grafo é uma representação simbólica de uma rede e da sua conectividade, em que todo o terminal e ponto de intersecção (real) se torna um nó, conectado a outro(s) por um segmento recto ou directo. Por outras palavras, um grafo é um par ordenado de conjuntos distintos (V, A), onde V é um conjunto arbitrário que se designa por conjunto de vértices ou nós, e A um conjunto de pares não ordenamentos de elementos (distintos) de V que se designa por conjunto de arestas.

i. No ensaio de modelação da susceptibilidade usaram-se sobretudo ferramentas de geoprocessamento de dados vectoriais, fazendo sentido tentar avaliar a existência e a performance de algoritmos especializados em trabalho com *raster*.

ii. A construção de um grafo aplicado a uma rede viária é uma tarefa morosa e árdua que deve respeitar todas as políticas de conectividade topológica, pelo que não foi possível preparar um grafo que garantisse esta exigência (devido à escassez de tempo para realização desta tarefa).

iii. Com os dados que tínhamos disponíveis, sempre que necessário conseguimos integrar as vias primárias sem haver necessidade de soluções complementares ou híbridas[195].

iv. Valorizámos também a viabilidade de incorporação do maior número possível de variáveis físicas que condicionam a deslocação.

Face ao exposto, a implementação do modelo de cálculo da acessibilidade a infra-estruturas de saúde consuma-se com a definição das seguintes tarefas ordenadas:

i. Preparação das variáveis que condicionam a mobilidade: infra-estruturas de partida ou de destino; declives; uso e ocupação do solo; barreiras (corpos de água e vias primárias); e rede viária (hierarquizada segundo os limites de velocidade);

ii. Criação de uma superfície de custo - operacionalização de uma cadeia lógica de processos que têm como fim estratificar o espaço em função do esforço, custo, resistência ou impedância necessária para percorrer cada célula da área de teste;

iii. Construção, a partir da superfície de custo, de uma nova superfície em que o valor de cada pixel expressa o custo acumulado de deslocação entre esse lugar e a unidade de saúde mais próxima. Este custo total representa o caminho mais curto (que envolve menos custo) entre os dois lugares;

[195] Ferreira (2012) propõe uma solução de compromisso entre os dois modelos de dados. O autor optou pela utilização do modelo matricial, mas, para resolver os problemas advindos da inclusão da rede de auto-estradas, recorre a uma base vectorial. O algoritmo de integração das vias primárias proposto por este autor consiste no cálculo iterativo, para cada nó de entrada na via rápida, do custo cumulativo entre cada célula e cada um dos outros nós, e cálculo da distância-custo entre cada nó e a infra-estrutura de interesse mais próxima, para posterior agregação num único cartograma. Faria todo o sentido utilizarmos um esquema conceptual deste género, visto que exige uma elevada capacidade de processamento por parte da máquina e do *software* utilizado (se existirem 70 nós na área de teste, este algoritmo criará 140 *rasters* e terá de os processar simultaneamente); porém, este procedimento obriga-nos a incluir no modelo um grafo da rede que respeite escrupulosamente todas as regras de conectividade topológica mas, também, as regras de trânsito.

iv. Quantificação do número aproximado de indivíduos que necessitam de um determinado intervalo de tempo para serem assistidos e transportados à unidade de saúde mais próxima em caso de emergência médica.

Uma vez identificadas todas as etapas/tarefas processuais do presente exercício, com o objectivo de averiguar a capacidade de manipulação de diferentes volumes de dados, estabeleceram-se quatro áreas de teste (Figura 63), sobre as quais foram executadas todas as etapas acima descritas.

I.I. Preparação das variáveis de entrada do modelo

O MDT foi gerado com recurso ao Programa 16[196], a partir da informação altimétrica e hidrográfica da Carta Militar de Portugal do Instituto Geográfico do Exército (série M888), mantendo-se a opção metodológica adoptada no exercício de cálculo da susceptibilidade, aquela que melhor se parece adequar aos dados vectoriais disponíveis.

A informação relativa ao uso e ocupação do solo foi extraída da Carta de Uso e Ocupação do Solo (COS) de 2007 disponibilizada pela DGT até ao segundo nível de desagregação.

A rede viária foi construída tendo por base a informação geográfica voluntária disponibilizada pelo *Open Street Maps*. Todas as linhas foram editadas, hierarquizadas (segundo o tipo e a velocidade) e validadas (por comparação com a realidade actual pela via da foto-observação e foto-interpretação de ortofotos).

Como barreiras (locais onde a circulação não é possível) foram definidas as áreas circundantes (*buffer* de 50 metros) às vias primárias (extraídas dos dados do *Open Street Maps*) e os corpos de água (COS 2007).

Todos estes temas foram preparados de modo a serem topologicamente coincidentes (polígonos) ou contidos (linhas) com a área de teste 4 (Figura 63).

Por fim, os serviços de saúde e as infra-estruturas que albergam os veículos de emergência, foram inventariados e georreferenciados, associando-se um conjunto específico de infra-estruturas a cada teste, tal como se sistematiza na Tabela XXXVI. O inventário destes equipamentos foi feito a partir da informação disponível em http://www.portaldasaude.pt/portal (acedido em 27 Abril, 2015), e o seu posicionamento foi realizado com o Google Earth.

[196] **Ver Anexo A, p. 13.**

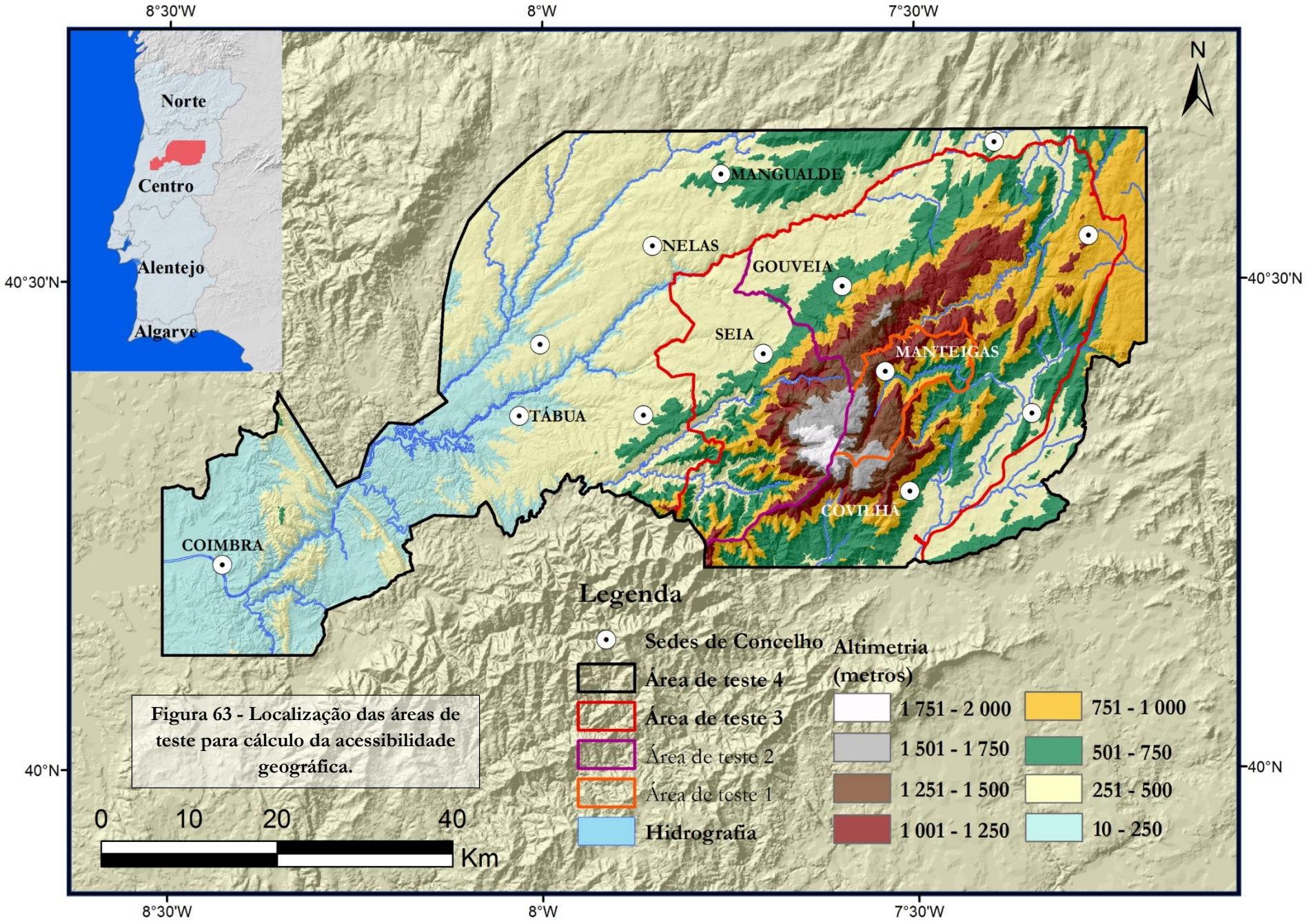


Figura 63 - Localização das áreas de teste para cálculo da acessibilidade geográfica.

Tabela XXXVI - Serviços de saúde e outras organizações que dispõem de meios para transporte de doentes

1	-» Centro de Saúde de Manteigas;	2	-» Centro de Saúde de Seia; -» Hospital Nossa Senhora da Assunção – Seia
	-» Bombeiros Voluntários de Manteigas.		-» Bombeiros Voluntários de Loriga; -» Bombeiros Voluntários de Seia; -» Bombeiros Voluntários de São Romão.
3	-» Hospital Pêro da Covilhã; -» Hospital Sousa Martins (Guarda); -» Bombeiros Voluntários de Belmonte; -» Bombeiros Voluntários de Celorico da Beira; -» Bombeiros Voluntários da Covilhã; -» Bombeiros Voluntários de Folgosinho; -» Bombeiros Voluntários de Fornos de Algodres; -» Bombeiros Voluntários da Guarda; -» Bombeiros Voluntários de Manteigas; -» Bombeiros Voluntários de Melo; -» Bombeiros Voluntários de Loriga; -» Bombeiros Voluntários de Seia; -» Bombeiros Voluntários de São Romão; -» Bombeiros Voluntários de Gonçalo; -» Bombeiros Voluntários de Unhais da Serra.	4	-» Instituto Português de Oncologia Francisco Gentil; -» Bombeiros Voluntários de Belmonte; -» Bombeiros Voluntários de Celorico da Beira; -» Bombeiros Voluntários da Covilhã; -» Bombeiros Voluntários de Folgosinho; -» Bombeiros Voluntários de Fornos de Algodres; -» Bombeiros Voluntários da Guarda; -» Bombeiros Voluntários de Manteigas; -» Bombeiros Voluntários de Melo; -» Bombeiros Voluntários de Loriga; -» Bombeiros Voluntários de Seia; -» Bombeiros Voluntários de São Romão; -» Bombeiros Voluntários de Gonçalo; -» Bombeiros Voluntários de Unhais da Serra.

1.2. 1ª Tarefa - Arquitectura do algoritmo que se destina a criar a superfície de custo

Os vários programas usados para produzir cartografia de isócronas têm o seu substrato conceptual e lógico no algoritmo de modelação da acessibilidade física apresentado por Ferreira (2012 e 2013). De acordo com a Figura 64, num primeiro momento, trata-se de um procedimento que atribui uma penalização, em tempo, a cada classe dos indicadores de custo (Tabela XXXVII), que expressa a menor ou maior facilidade de locomoção pelo espaço. A agregação de todos os temas matriciais (e respectivas penalizações associadas a cada célula) consuma o processo de construção da superfície de custo.

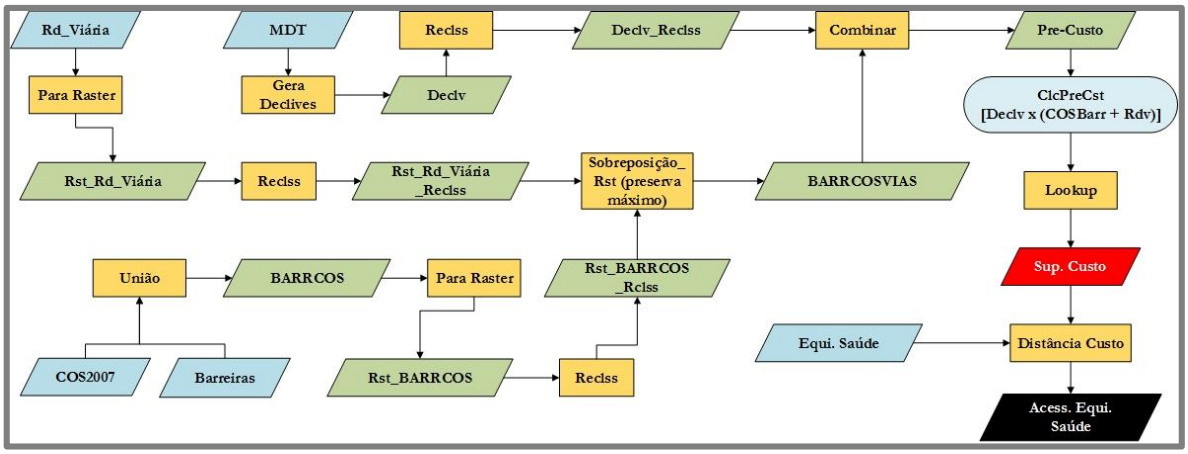


Figura 64 - Modelo lógico do algoritmo de modelação da acessibilidade física proposto por Ferreira (2012 e 2013), inspirado no modelo conceptual de Julião (2001, p. 211).

Tabela XXXVII - Penalizações em tempo (segundos) atribuídas a cada classe de declives, uso e ocupação do solo e rodovias. Adaptado de Ferreira (2012 e 2013)

Declive (%)	Peso "fora da estrada"	Peso rodovias[197]
0 – 10	1	1
10 – 30	1,5	1
30 – 50	2	1,5
50 – 70	3	1,5
70 – 100	4	2
> 100	5	2
Circulação "fora da estrada"		
Classe COS	**Velocidade Média[198]**	**Tempo (resol. 10 m)[199]**
Floresta, matos e incultos	25 min./Km	15 Segundos
Terrenos sem cobertura	20 min./Km	12 Segundos
Áreas húmidas	38 min./Km	23 Segundos
Agricultado	30 min./Km	18 Segundos
Urbano	15 min./Km	9 Segundos
Corpos de água /Barreiras	NoData	-
Circulação nas rodovias		
Tipo de via	**Veloc. Média**	**Tempo (resol. 10 m)[200]**
Cruzamentos/Rotundas	10 Km/h	3,60 Segundos
Cruzamentos/Rotundas	15 Km/h	2,70 Segundos
Cruzamentos/Rotundas	20 Km/h	1,80 Segundos
Rua/Caminho	25 Km/h	1,44 Segundos
Rua/Acesso/Praça/Caminho	30 Km/h	1,20 Segundos

[197] Esta coluna corresponde a um refinamento dos pesos, resultante do processo de calibração do modelo. Tendo em conta que os valores de declives são calculados a partir das curvas de nível das cartas da série M888, sem se ter em consideração as obras de engenharia efectuadas para nivelar as áreas onde são implantadas as estradas, optou-se por diferenciar as penalidades impostas nas células que correspondem a vias das que dizem respeito a espaços não abrangidos por elas.

[198] Assim definida por Ferreira (2012 e 2013).

[199] Definida pela expressão: $tempo = \frac{(minutos \times 60) \times resolução\ da\ célula}{1000}$.

[200] Calculada segundo a fórmula: $tempo = \frac{resolução\ da\ célula \times 3600}{velocidade \times 1000}$

Rua	35 Km/h	1,03 Segundos
Rua	40 Km/h	0,90 Segundos
Rua/Estrada	45 Km/h	0,80 Segundos
Rua/Estrada	50 Km/h	0,72 Segundos
Estrada	55 Km/h	0,65 Segundos
Estrada	60 Km/h	0,60 Segundos
Estrada Nacional	65 Km/h	0,55 Segundos
Estrada Nacional	70 Km/h	0,51 Segundos
Estrada Nacional	75 Km/h	0,48 Segundos
Estrada Nacional	80 Km/h	0,45 Segundos
Estrada Nacional	90 Km/h	0,40 Segundos
Itinerários principais e complementares	100 Km/h	0,36 Segundos
Auto-estradas	120 Km/h	0,30 Segundos

Este modelo prevê a entrada das vias primárias como barreira, sendo excluídas do tema que representa a rede viária. No presente exercício optámos por introduzir uma alteração, que tem precisamente a ver com o que dissemos anteriormente sobre resolver o problema das falsas intersecções num grafo planar. Como se pode ver na Figura 63, a área de teste 4 abrange uma extensão que vai desde o Parque Natural da Serra da Estrela (PNSE) até ao Baixo Mondego, nomeadamente Coimbra, tendo sido especificamente recortada para avaliar a distância-tempo entre o PNSE e infra-estruturas prestadoras de serviços de saúde altamente especializados como é o caso do Instituto Português de Oncologia de Coimbra (cfr. Tabela XXXVI). Uma vez que a deslocação até Coimbra por via rápida é menos morosa, este é o único ensaio que justifica a integração deste tipo de vias na análise. Neste exercício específico, o nosso objectivo é saber quanto tempo os indivíduos que residem no PNSE demoram a deslocar-se até Coimbra, portanto, a rede viária exterior aos limites da área de teste 3 contempla apenas os troços mais utilizados neste movimento (por possibilitarem uma viagem mais rápida), entre os quais se incluem, para além da Nacional 17, a A-23, a A-25, o IP-3, o IC-6 e o IC-12 (bem como os seus acessos). Consequentemente deixam de constar quaisquer intersecções planas que não existem na realidade.

Com esta opção tomada, para a área de teste 4, em específico, incluímos as vias primárias no tema da rede rodoviária, mas não deixamos de as integrar também como barreiras. Esta abordagem suscita a criação de corredores de circulação (Figura 65) que garantem a consistência e validade do processo, uma vez que o *software* só considera como válidas as conexões reais entre a rede secundária e a primária, pois só estas intersecções é que têm um valor diferente de *NoData*.

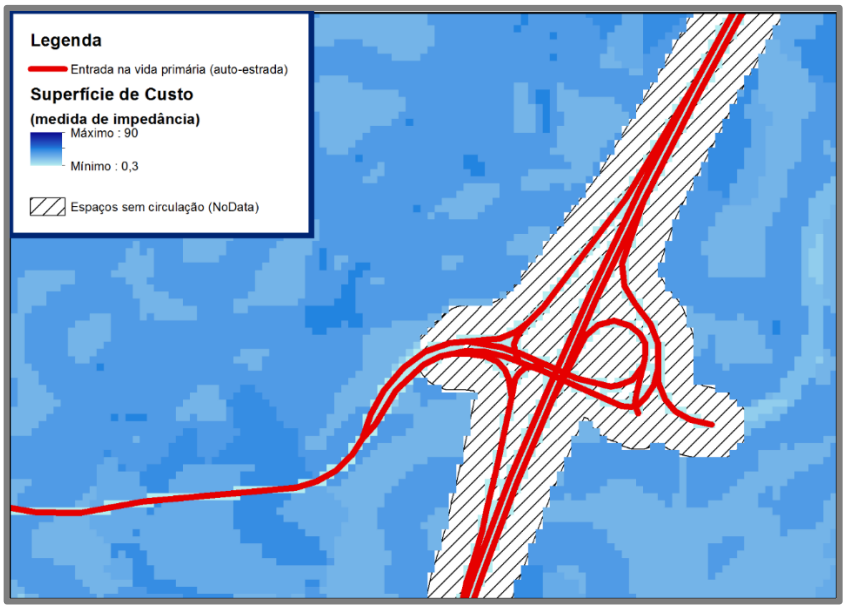


Figura 65 - Exemplo de um corredor de circulação numa via primária (a via primária é circundada por valores *NoData*, de modo a que só seja possível aceder a ela nas intersecções reais com as vias secundárias).

No entanto, ainda que possamos considerar esta alternativa de indução do caminho que o *software* deve tomar, ela não deixa de ser uma abordagem improvisada e adequada apenas a este problema concreto, pois, se quiséssemos avaliar as impedâncias da área exterior à área de teste 3, esta técnica seria pouco razoável, na medida em que a rede considerada não é a real.

1.3. 2ª Tarefa - Criação de superfícies de custo acumulado

Tendo uma superfície de custo que expressa a resistência de cada célula ao movimento, a determinação da acessibilidade é concretizada com base num (ou em vários) ponto de referência (origem – partida ou chegada), a partir do qual é calculado, cumulativamente, o menor valor de custo de movimento no espaço (a partir da superfície de custo), tendo em conta as direcções cardeais e colaterais (Figura 66).

Nesta fase, optámos por testar as ferramentas dos vários *software* que implementam este algoritmo isoladamente, devido a que por si só, ele exige uma grande capacidade de processamento, na medida em que a atribuição de um valor de impedância acumulada a cada célula exige a determinação de todos os caminhos possíveis, devendo ser seleccionado posteriormente aquele que representa o menor custo acumulado.

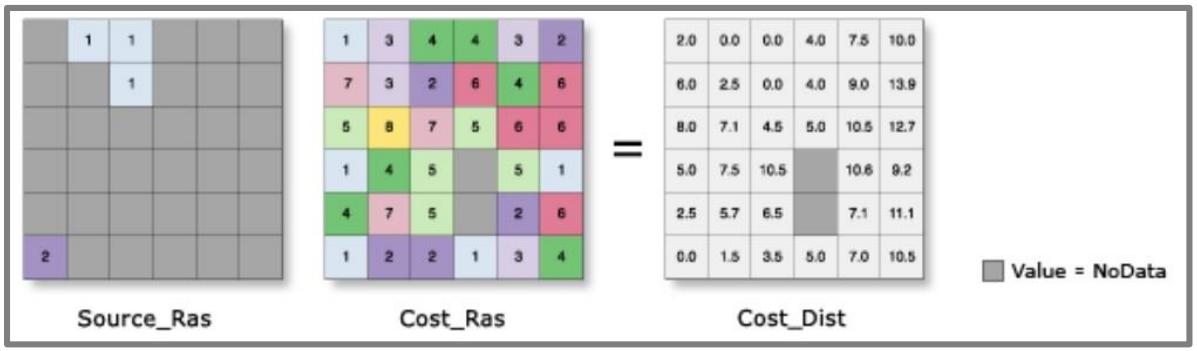


Figura 66 - Cálculo do custo acumulado entre cada pixel da área de estudo e os locais de partida/destino. Fonte: http://resources.arcgis.com/EN/HELP/MAIN/10.2/index.html#//009z00000018000000 (acedido em 12 Novembro, 2015).

Neste exercício definimos como *features* de destino as unidades de saúde inventariadas e atribuídas a cada área de teste.

1.4. 3ª Tarefa - Avaliação da capacidade de resposta em situações de emergência e quantificação do número de indivíduos que precisam de esperar mais tempo para receber socorro

Genericamente, esta última tarefa de teste teve como desígnio apurar o comportamento do *software* quando, numa única rotina, incluíssemos simultaneamente uma ferramenta como a de cálculo de impedância acumulada e outras comummente utilizadas (como a calculadora *raster*, reclassificação, conversão para formato vectorial, agregação, intersecção, entre outras).

Neste sentido, criou-se um algoritmo que implementa a cadeia lógica de geoprocessos apresentada na Figura 67, com o intuito de criar um cartograma de isócronas em que cada pixel representa o tempo desde a saída de uma ambulância até à sua chegada ao centro de saúde ou hospital mais próximo, passando por essa mesma célula (local onde se registou um pedido de auxílio), a partir do qual se quantifica o número de indivíduos que se intersectam com cada classe de impedância previamente definida.

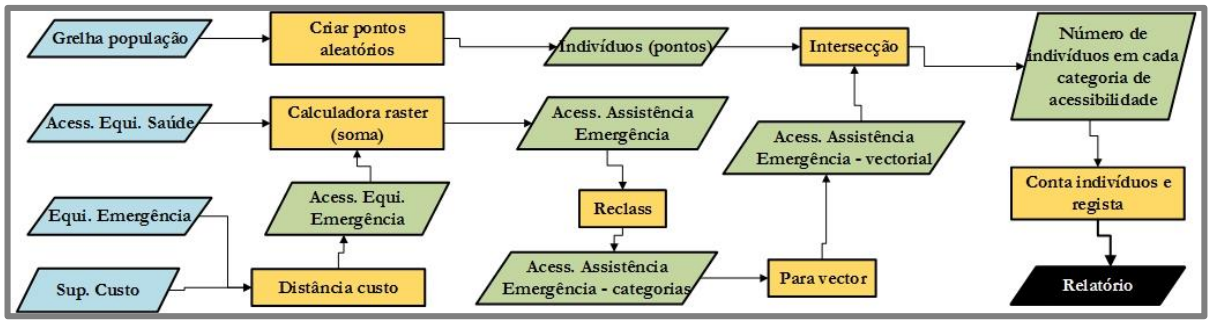


Figura 67 - Cadeia processual que tem como finalidade quantificar o número de indivíduos que se intersectam com cada categoria de impedância.

Para este último ponto em específico, recolheram-se os quantitativos populacionais para a área de teste 3. Estes valores encontram-se distribuídos por um conjunto de células com 1 Km^2 que se intersectam com as áreas em estudo[201]. A intersecção destes dados com as classes de acessibilidade implica a criação de pontos aleatórios, que simbolizam cada um dos indivíduos residentes em cada célula da grelha. A não utilização deste procedimento poderia enviesar os resultados, na medida em que uma determinada célula da grelha da população pode intersectar-se com mais do que uma classe de acessibilidade: se essa célula tiver 100 habitantes, a intersecção com uma das classes, associar-se-lhe-ia esse tanto; mais tarde, com a intersecção da mesma célula com outra classe, produziria o mesmo efeito, associando a essa classe, indivíduos já contabilizados, o que implicaria que, no final, iríamos obter um somatório cuja tradução cairia na impossibilidade de representar um valor superior ao total populacional da área considerada.

A simulação aleatória da distribuição da população pelas várias células desta grelha, criando-se, em cada uma delas, tantos pontos quanto o número de indivíduos que, segundo o INE, nela residem, é um procedimento que garante que o total de indivíduos contabilizados seja igual ao valor total real, e, embora seja aleatória, não é totalmente despropositada, visto que as pessoas, no seu dia-a-dia deslocam-se e podem não estar na sua habitação no momento em que se regista um evento que os obriga a solicitar mecanismos de assistência médica.

[201] Informação disponibilizada pelo Instituto Nacional de Estatística em http://geogrid.ine.pt (acedido em 20 Julho, 2015).

2. Base analítica para selecção de *software* com funcionalidades avançadas de manipulação de dados matriciais para implementação do modelo de cálculo da acessibilidade geográfica em ambiente SIG

> *"É importante reconhecer que grande parte dos projectos concentram-se no desenvolvimento de software com uma limitada gama de funcionalidades. Consequentemente, nem sempre é possível executar a totalidade de uma cadeia de processos com apenas uma aplicação SIG livre e/ou de código aberto."*
>
> Stefan Steiniger & Andrew Hunter (2013:139)[202]

2.1. Critérios que definem as soluções SIG a serem testadas

A AMC desenvolvida no capítulo anterior revelou como é difícil encontrar um *software* com todas as funcionalidades necessárias para a execução de um conjunto de etapas que consumam a concretização de uma meta de um investigador ou organismo. Na realidade, descobrimos, à semelhança de Steiniger & Hunter (2013), que um utilizador de SL/CA SIG é frequentemente obrigado a recorrer a vários *software* para completar a cadeia processual desenhada para responder aos objectivos estipulados.

Os resultados do Exercício 1 (Capítulo II da 2ª Parte) impelem-nos a concluir isso mesmo. Evitamos propositadamente a interoperabilidade entre os diferentes *software* SIG, isso, em alguns momentos, traduziu-se num incumprimento das metas. Assim, no presente exercício, admitimos maior flexibilidade, ou seja, no caso de falhar a ferramenta X do *software* Z, implementaremos uma solução que a corrigirá, mesmo que tenhamos de recorrer a recursos de outros pacotes, de modo a garantir que os objectivos idealizados sejam satisfeitos.

[202] No original, *"It is important to recognize that each of these projects concentrates its software development activities on a limited range of functionality. Hence, a certain set of GIS tasks may be accomplished easily with on FOSS4G application, whereas another FOSS4G solution may not provide the functionality due*" *In* STEINIGER, Stefan & HUNTER, Andrew (2013) – "The 2012 free and open source GIS *software* map – A guide to facilitate research, development, and adoption". **Computers, Environment and Urban Systems**, N.º 39, 136-150.

Por sua vez, na pré-selecção do *software* a testar, mantivemos a estrutura da AMC usada no primeiro exercício (cfr. Figura 52, p. 175), excepção feita para o Atributo 10, que inclui na análise um factor relacionado com experiências passadas, onde as várias alternativas são valoradas segundo a fiabilidade e performance que demonstraram no exercício anterior.

Aproveitámos também esta oportunidade para substituir as alternativas que obtiveram *scores* mais baixos, por outras que não foram trabalhadas[203]. Assim, na Tabela XXXVIII apresentamos as alternativas que iremos avaliar, de modo a aferir a sua predisposição relativamente à concretização da seguinte meta: eleição dos *software* com maior potencialidade para produzir cartografia de acessibilidade a unidades de saúde em situações de emergência.

Tabela XXXVIII - *Software* eleito para avaliação de predisposição para a concretização da meta estabelecida

0	Diva-GIS	5	Kosmo
1	FlowMap	6	Orbis GIS
2	GDAL	7	QGIS
3	GRASS GIS	8	SAGA GIS
4	Kalypso	9	Sextante integrado no gvSIG

Definidas as alternativas, procedemos à valoração e ponderação dos vários atributos, a Tabela XXXIX expressa os resultados dessa avaliação por pares. De um modo geral, quanto à estratégia de valoração e ponderação, respeitaram-se as adjectivações elucidadas anteriormente (AMC do Capítulo II da 2ª Parte). Ainda assim, a nova distribuição dos pesos pelos vários atributos resulta da inclusão do Atributo 10, da desvalorização do Atributo 3 e consequente sobrevalorização do Atributo 1. Esta alteração substancial na ponderação dos critérios justifica-se porque a associação do projecto à OSGeo não se traduz necessariamente numa capacidade de concorrer directamente com o SP, constatação que resulta do Exercício 1 (Capítulo II da 2ª Parte).

Tabela XXXIX - Valoração e ponderação dos vários atributos e *score* final que indica a predisposição das alternativas relativamente à meta estipulada – segundo o PAH[204]

Soft.		Atr.1	Atr.2	Atr.3	Atr.4	Atr.5	Atr.6	Atr.7	Atr.8	Atr.9	Atr.10	Score
0	S	0,05	0,04	0,03	0,06	0,12	0,02	0,03	0,02	0,03	0,04	
	S_p	1,1	0,5	0,2	1,3	0,4	0,1	0,1	0,1	0,4	0,5	**4,6**
1	S	0,03	0,03	0,03	0,02	0,02	0,18	0,03	0,02	0,03	0,04	

[203] A AMC que levamos a cabo no capítulo anterior obrigou-nos a estudar superficialmente todos os *software* envolvidos, pois constatámos que estes *software* (que agora são excluídos) dificilmente conseguiriam competir com soluções mais aprimoradas, como é o caso do GRASS GIS.

[204] Nesta tabela, S é a valoração atribuída a essa alternativa, tendo em conta determinado atributo; por sua vez S_p é o produto da multiplicação de S por 100 e pelo peso que esse atributo tem na tomada de decisão.

	S_p	0,7	0,4	0,2	0,4	0,1	0,5	0,1	0,1	0,4	0,5	**1,3**
2	S	0,05	0,07	0,26	0,24	0,02	0,18	0,03	0,14	0,26	0,20	
	S_p	1,1	0,9	1,3	5,3	0,1	0,5	0,1	0,7	3,4	2,6	**16,0**
3	S	0,34	0,35	0,26	0,24	0,20	0,18	0,26	0,14	0,26	0,42	
	S_p	7,5	4,6	1,3	5,3	0,6	0,5	0,8	0,7	3,4	5,5	**30,1**
4	S	0,03	0,03	0,03	0,02	0,02	0,02	0,03	0,02	0,03	0,04	
	S_p	0,7	0,4	0,2	0,4	0,1	0,1	0,1	0,1	0,4	0,5	**2,9**
5	S	0,03	0,03	0,03	0,02	0,12	0,02	0,03	0,14	0,03	0,04	
	S_p	0,7	0,4	0,2	0,4	0,4	0,1	0,1	0,7	0,4	0,5	**3,8**
6	S	0,03	0,03	0,03	0,03	0,02	0,02	0,03	0,14	0,03	0,04	
	S_p	0,7	0,4	0,2	0,7	0,1	0,1	0,1	0,7	0,4	0,5	**3,7**
7	S	0,10	0,10	0,26	0,24	0,20	0,18	0,26	0,14	0,26	0,11	
	S_p	2,2	1,3	1,3	5,3	0,6	0,5	0,8	0,7	3,4	1,4	**17,5**
8	S	0,10	0,13	0,03	0,06	0,05	0,18	0,03	0,14	0,03	0,04	
	S_p	2,2	1,7	0,2	1,3	0,2	0,5	0,1	0,7	0,4	0,5	**7,8**
9	S	0,26	0,21	0,03	0,06	0,20 [205]	0,02	0,26	0,14	0,03	0,04	
	S_p	5,7	2,7	0,2	1,3	0,6	0,1	0,8	0,7	0,4	0,5	**13,0**
RC		**2%**	**3%**	**0%**	**2%**	**1%**	**0%**	**0%**	**0%**	**0%**	**2%**	
Peso		**0,22**	**0,13**	**0,05**	**0,22**	**0,03**	**0,03**	**0,03**	**0,05**	**0,13**	**0,13**	

No entanto, as mudanças introduzidas não potenciaram resultados muito divergentes dos da Tabela XXIX. A biblioteca GDAL/OGR, o GRASS GIS e o QGIS foram mais uma vez os melhores classificados, facto que acaba por denunciar uma certa imaturidade ou especialização das outras distribuições[206].

Entre os três melhores classificados, o GRASS destaca-se novamente, sobretudo, por disponibilizar todas as ferramentas indispensáveis ao desenvolvimento do processo, bem como, de quase todas as ferramentas complementares e que constam da Tabela XL. Este *software* encontra-se num restrito grupo de distribuições livres (onde se inserem também o SAGA GIS e a Sextante) com ferramentas específicas para modelação da acessibilidade, nomeadamente, aquela que efectua o cálculo do custo acumulado.

Tabela XL - Ferramentas que os *software* devem ter para executar o procedimento de modelação da acessibilidade geográfica

Ferramentas indispensáveis	Ferramentas complementares
-» Gerar matriz de declives; -» Reclassificação de ficheiros matriciais; -» Edição de dados tabulares; -» União de ficheiros vectoriais;	-» *Lookup;* -» Soma ponderada; -» Corte topológico; -» Criação de pontos aleatórios;

[205] Neste atributo, o Sextante aparece melhor cotado (mais amigável) do que na AMC do exercício 1 porque aí consideramo-la individualmente, nesta nova análise, pelo contrário, consideramo-la integrada no gvSIG.
[206] Por exemplo, o Kalypso é especializado na análise de dados geográficos no âmbito da problemática do risco de inundação e da modelação de redes hidráulicas; por seu turno, o DIVA-GIS foca-se na análise de dados climáticos.

-» Conversão *Vector To Raster*; -» Sobreposição de dois ficheiros matriciais, preservando o valor máximo entre cada par de pixéis (Figura 68); -» Combinar dois ficheiros matriciais, estipulando categorias que representam todas as combinações de valores entre as duas imagens; -» Intersecção; -» Cálculo do custo acumulado entre dois pontos.	-» Agregação (*dissolve*); -» Extracção (*Select > Export*); -» *Split;* -» Calculadora *raster*.

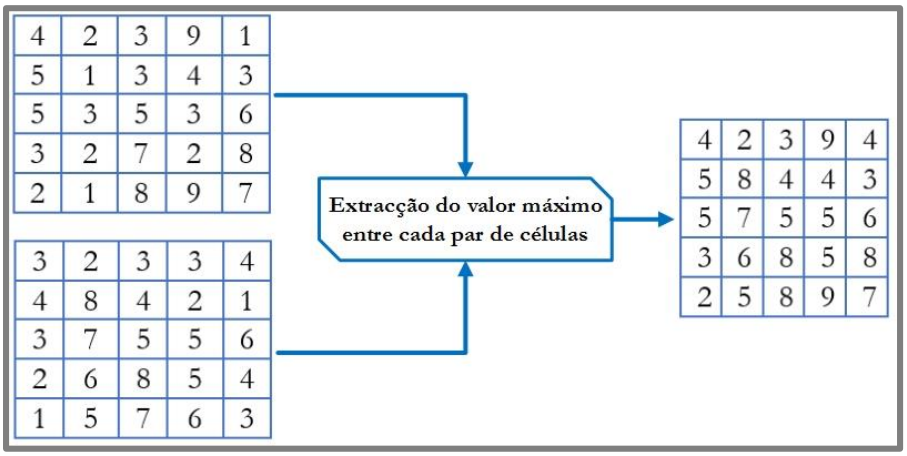


Figura 68 - Sobreposição entre duas matrizes, da qual resulta uma nova imagem cujos valores dizem respeito ao valor máximo resultante da comparação entre cada par de células.

Apesar dos bons resultados obtidos, nem o QGIS nem a GDAL/OGR dispõem deste tipo de algoritmos, o que justifica as diferenças mais expressivas da Tabela XXXIX quando confrontado com a Tabela XXIX. Consequentemente, pareceu-nos ser avisada a opção de passar a uma fase de confronto o ArcGIS e o GRASS GIS para realização destes exercícios.

2.2. Características dos dados de entrada e propriedades das rotinas que implementam os algoritmos de cálculo da acessibilidade a infra-estruturas de saúde

Eleitas as alternativas a comparar, e sabendo que a performance e o desempenho do *software* variam com as características e volume dos dados de entrada, destacamos, na Tabela XLI, as propriedades mais relevantes dos dados de entrada.

Tabela XLI - Características dos dados de entrada do exercício de modelação da acessibilidade geográfica (cfr. Figura 63, p. 215 para identificação das áreas de teste)

Área teste	Tema	*Features*/Células (10x10)	Memória em disco (Kb)	Área (m^2) incluindo valores NoData
1ª Tarefa – Criação da superfície de custo				
1	MDT (*raster*)	1 219736	4 614	2 753 560 x 100
	COS (vector)	331	925	
	Rede viária (vector)	545	194	
	Barreiras (vector)	1	146	
2	MDT	4 356 881	15 452	7 677 220 x 100
	COS	2 398	3 847	
	Rede viária	2 782	709	
	Barreiras	1	156	
3	MDT	17 460 131	61 538	26 254 200 x 100
	COS	9 418	17 472	
	Rede viária	12 236	3 136	
	Barreiras	1	919	
4	MDT	40 654 252	136 224	67 222 400 x 100
	COS	29 657	45 052	
	Rede viária	22 169	4 616	
	Barreiras	1	2 368	
2ª Tarefa – Criação de superfícies de custo acumulado				
1	Unidades saúde	1	1	2 753 560 x 100
	Superfície de custo	1 219736	11 653	
2	Unidades saúde	2	1	7 677 220 x 100
	Superfície de custo	4 356 881	31 114	
3	Unidades saúde	2	1	26 254 200 x 100
	Superfície de custo	17 460 131	104 860	
4	Unidades saúde	1	1	67 222 400 x 100
	Superfície de custo	40 654 252	264 771	
3ª Tarefa – Avaliação da capacidade de resposta em situações de emergência e quantificação do número de indivíduos que precisam de esperar mais tempo para receber socorro				
1	Equipamentos emergência	1	1	2 753 560 x 100
	Custo acumulado - saúde	1 219736	11 653	
	Superfície de custo	1 219736	11 653	
	Grelha população (vector)	159	55	
2	Equipamentos emergência	3	1	7 677 220 x 100
	Custo acumulado - saúde	4 356 881	31 113	
	Superfície de custo	4 356 881	31 114	
	Grelha população	429	79	
3	Equipamentos emergência	13	1	26 254 200 x 100
	Custo acumulado - saúde	17 460 131	104 859	
	Superfície de custo	17 460 131	104 860	
	Grelha população	1 622	216	
4	Equipamentos emergência	13	1	67 222 400 x 100
	Custo acumulado - saúde	6 088/40 654 252	264 770	
	Superfície de custo	40 654 252	264 771	
	Grelha população	1 622	216	

Estes dados são processados pelos programas que materializam os algoritmos conceptuais referidos no primeiro tópico deste capítulo e que estabelecem cada uma das tarefas (ArcGIS e GRASS GIS). As suas principais características constam da Tabela XLII.

Tabela XLII - Principais características dos procedimentos construídos e implementados para a produção de cartografia de acessibilidade a serviços/unidades de saúde

Software	Prog.	Linhas	Nº de Geoprocessos usados	N.º Ferramentas nativas	N.º Ferramentas de outras distribuições	N.º Ferramentas criadas	*Outputs* intermédios (Kb) - máx	Complexidade
1ª Tarefa – Criação da superfície de custo								
ArcGIS	40	193	11	11	0	0	432 000	1
GRASS	41	237	10		0	0	728 000	1
2ª Tarefa – Criação de superfícies de custo acumulado								
ArcGIS	42	20	1	1	0	0	0	1
GRASS	43	22	1	1	0	0	0	1
3ª Tarefa – Avaliação da capacidade de resposta em situações de emergência e quantificação do número de indivíduos que precisam de esperar mais para receber socorro								
ArcGIS	44	191	10	10	0	0	1 160 743	1
GRASS	45	209	10	9	1	1	?	3

Em termos de cadeia processual de geoprocessamento, estes programas implementam os seguintes procedimentos lógicos:

i. **1ª Tarefa (cfr. Figura 64, p. 217, Programas 40[207] e 41[208]) -** Com estes programas experimentamos algumas ferramentas que têm em vista a integração/associação de vários ficheiros matriciais num só, segundo um conjunto de operadores e regras lógicas. Para além disso, estas rotinas incluem métodos para edição de dados tabulares, visando a atribuição de ponderações a cada categoria de um tema, sendo esse peso representativo do esforço necessário para atravessar a célula. Da multiplicação destes valores ponderados resulta o custo associado ao movimento na célula. Isto é conseguido segundo a seguinte sequência lógica de etapas:

   a. Cálculo da carta de declives e sua reclassificação (cfr. Tabela XXXVII, p. 217);

   b. Agregação das várias classes de uso e ocupação (nível 2) em classes generalistas (cfr. Tabela XXXVII, p. 217);

   c. União da COS e das barreiras, e posterior conversão para *raster*;

[207] **Ver Anexo A, p. A66.**
[208] **Ver Anexo A, p. A69.**

d. Conversão da rede viária (modelo de dados vectorial) para modelo de dados matricial;

e. Sobreposição da rede viária com o *raster* COS/Barreiras, usando uma ferramenta que concretiza o ilustrado no esquema da Figura 68[209].

f. Combinação deste *output* intermédio com os declives reclassificados (*Combine*/*r.report*). Os programas lêem os dados tabulares resultantes e, para cada combinação possível de valores (combinação entre categorias de declive e categorias de velocidade/categorias de uso e ocupação do solo), estimam o custo total de movimento na célula.

ii. **2ª Tarefa (cfr. Figura 66, p. 220, Programas 42[210] e 43[211]) –** Estas instruções executam a ferramenta *Cost Distance* (ArcGIS 10.2) e *r.cost* (GRASS GIS 7.0.0), algoritmos que calculam o custo acumulado do movimento no espaço entre cada célula e um (ou vários) ponto de referência (unidade de saúde).

iii. **3ª Tarefa (cfr. Figura 67, p. 221, Programas 44[212] e 45[213]) –** Com a finalidade de apurar o comportamento do *software* quando incluímos na mesma cadeia de geoprocessos uma ferramenta que exige muito da máquina (tal como o *Cost Distance*/*r.cost*) e outras frequentemente utilizadas, estes procedimentos seguem a seguinte lógica processual:

a. Criação de uma grelha de isócronas entre cada célula e as infra-estruturas que albergam os equipamentos de resposta em situações de emergência médica;

b. Soma desta última com o resultado dos Programas 42 e 43, originando uma superfície cujos pixéis representam a distância-tempo entre os equipamentos que dão resposta a emergências e a unidade de saúde mais próxima a partir do local onde foi solicitado auxílio;

c. Categorização desta matriz em classes de acessibilidade;

---

[209] A preservação do valor máximo faz com que no resultado se mantenham representados os pixéis com valores de velocidade onde existir asfalto, pois estes valores são superiores aos números que identificam as classes de uso e ocupação do solo. Onde não existem estradas, o valor do pixel será 0, portanto, na combinação com o COS/Barreiras, a ferramenta vai preservar o identificador da classe de ocupação do solo.

[210] **Ver Anexo A, p. 73.**

[211] **Ver Anexo A, p. 74.**

[212] **Ver Anexo A, p. 74.**

[213] **Ver Anexo A, p. 78.**

d. Criação de pontos aleatórios tendo por base a grelha que fornece dados sobre os quantitativos populacionais associados a cada um dos seus pixéis – cada ponto diz respeito a um indivíduo;

e. Conversão da matriz hierarquizada segundo as várias categorias de acessibilidade para o modelo de dados vectorial e intersecção com os pontos que representam a população, registando-se, iterativamente, o número de indivíduos que se intersectam com cada classe de impedância que nos revela o tempo de espera para pessoas serem assistidas e levadas à unidade de saúde mais próxima.

3. Comparação de desempenho entre *Software* SIG Proprietário e *Software* SIG Livre: 2ª Análise

> *"O GRASS foi escrito em módulos, tendo em vista a minimização da sobrecarga. Isto permite aos utilizadores correrem o sistema, ou partes dele, em pequenas máquinas portáteis com RAM limitada, ou a escalá-lo para processar grandes conjuntos de dados que excedam a memória do sistema disponível".*
>
> Markus Neteler, Hamish Bowman, Martin Landa & Markus Met (2012:127)[214]

Neste segundo momento de análise comparada da resposta dos *software* seleccionados perante um problema concreto e específico, comparamos o desempenho e a performance do ArcGIS 10.2 e GRASS GIS 7.0.0 no âmbito da produção de cartografia de acessibilidade a serviços/unidades de saúde, recorrendo a três exercícios/tarefas que acabamos que enunciar.

Na realização da 1ª Tarefa (criação da superfície de custo), de acordo com o gráfico da Figura 69[215], o desempenho dos dois *software* em confronto foi bastante equilibrado, uma vez que não se podem assinalar assimetrias significativas nos tempos registados, ao contrário de tudo o que até aqui se tinha verificado. Não obstante, o ArcGIS 10.2 foi sempre o mais rápido. Por sua vez, o GRASS GIS, para além de ter sido o menos célere, a sua capacidade de processamento reflecte, de modo mais evidente, o impacto que o aumento do volume de dados tem, pois, tal como é visível na Figura 69, à medida que o volume de dados aumenta, as diferenças entre o tempo registado nessa iteração e o tempo verificado na iteração precedente é mais notório do que o ArcGIS, que, pelo contrário,

[214] No original, "*GRASS is written in a fully modular way which minimizes over-head. This allows users to run the system, or parts of it, even in portable smart devices with limited RAM, or effectively scale it up to process massive datasets which exceed available system memory by orders of magnitudes.*" *In* NETELER, Markus; BOWMAN, Hamish; LANDA, Martin & METZ, Markus (2012) – "GRASS GIS: A multi-purpose open source GIS". **Environmental Modelling & *Software***, N.º 31, 124-130.

[215]

	2 753 560 Células	7 677 220 Células	26 254 000 Células	67 222 400 Células
ArcGIS 10.2	00:00:08	00:00:15	00:00:45	00:02:20
GRASS GIS 7.0.0	00:00:13	00:00:36	00:01:52	00:05:42

parece suportar melhor o aumento do volume de dados, facto que vem sublinhar uma diferença apreciável entre os comportamentos dos dois software.

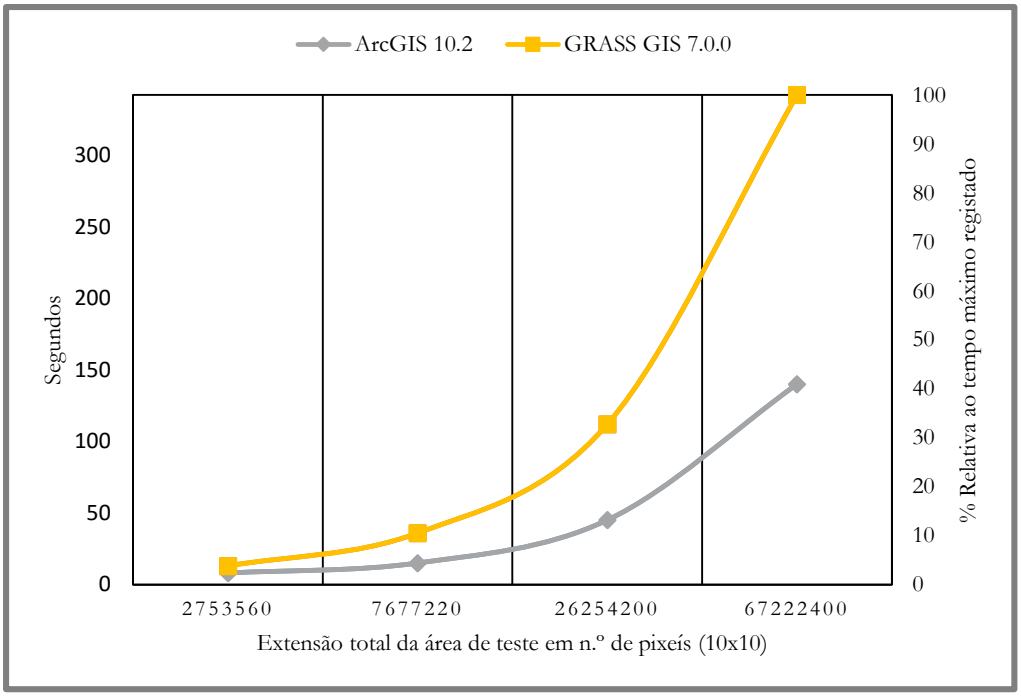


Figura 69 - Tempo consumido na execução da tarefa de criação da superfície de custo, tendo em conta o aumento do volume dos dados de entrada.

Estamos conscientes de que as possíveis razões para a explicação destes resultados poderão ser francamente subjectivas e, até, discutíveis. Podem ter origem na construção do código-fonte das ferramentas utilizadas, ou podem ter a ver com a obrigatória conversão entre formatos quando pretendemos trabalhar com imagens *.tif* e vectores *.shp* no GRASS (aproveitamos para lembrar que este *software* só processa dados que estejam em formato *grass vector* ou *grass raster*). Acresce que, o facto de os dados temporários gerados ao longo do processo serem mais pesados quando comparados com os que foram criados pelo ArcGIS, implicarem um impacto maior sobre a memória (cfr. Tabela XLII, p. 228), facto que poderá revestir também uma hipótese explicativa para a diferença de comportamentos entre os *software* em confronto.

Em termos gráficos/visualização (Figura 70), a cartografia gerada pelos Programas 40[216] e 41[217] apresenta muitas semelhanças em termos de distribuição espacial dos valores de impedância.

[216] **Ver Anexo A, p. A66.**
[217] **Ver Anexo A, p. A69.**

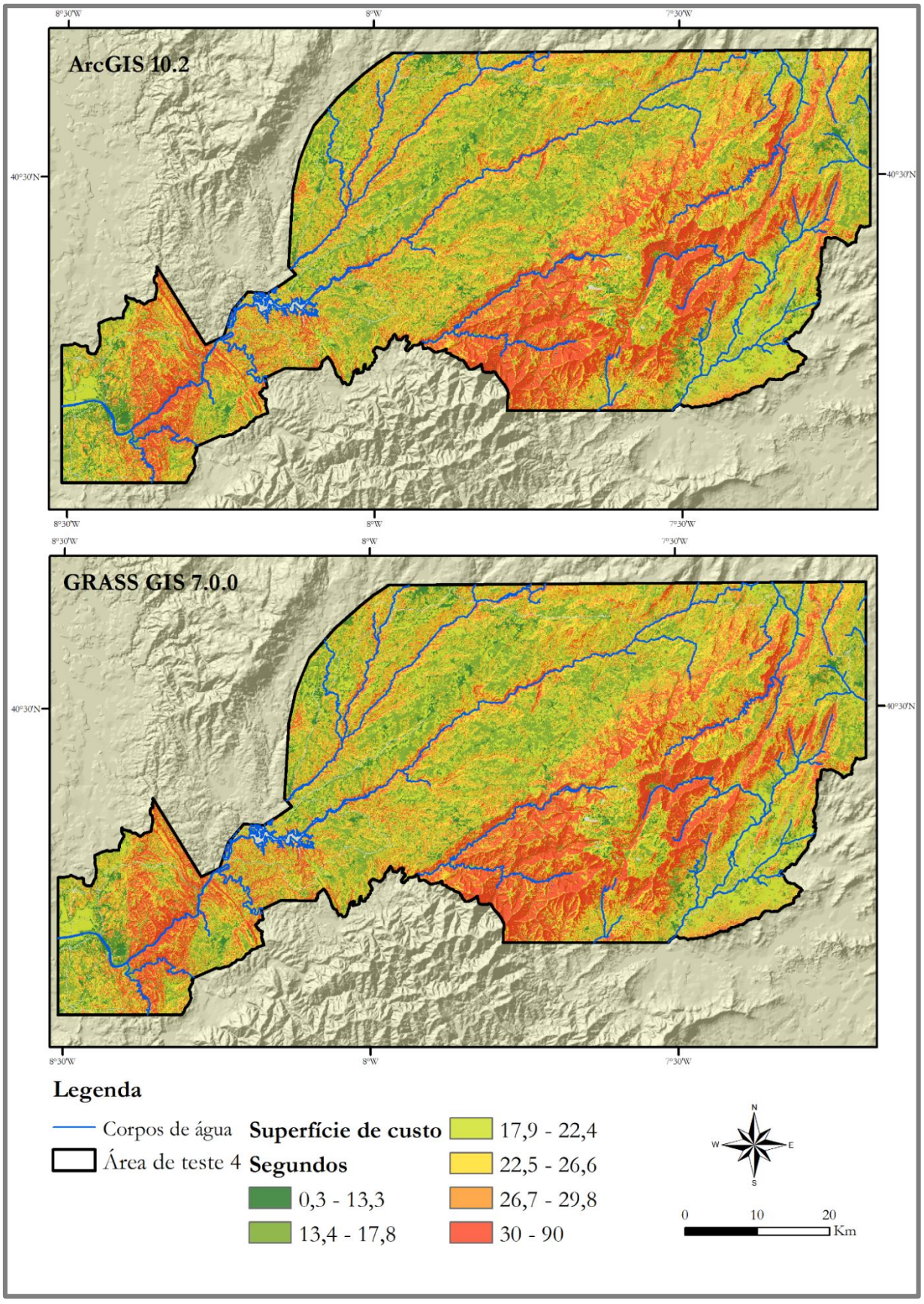


Figura 70 - Superfícies de custo obtidas mediante a execução dos Programas 40 e 41.

Porém, uma análise do suporte estatístico destes dados, apesar de revelar uma correlação acentuada entre as duas imagens matriciais (99%), alerta-nos para a existência de pequenas distorções entre os dois *outputs*, que, em princípio, deveriam ser exactamente iguais, na medida em que construímos os dois programas em questão de modo a produzirem resultados perfeitamente iguais.

Apesar da similaridade dos Programas 40 e 41, que, ao contrário de outros, empregam em exclusivo ferramentas análogas, a execução do Programa 12[218] identifica 154 349 (0,90%) pixéis com valores distintos nas duas imagens. Estas distorções, aparentemente inexplicáveis, têm a sua razão de ser nas diferenças entre a ferramenta *Polyline To Raster* do ArcGIS e a ferramenta análoga (*v.to.raster*) do GRASS GIS. Como se pode observar na Figura 71, as diferenças ocorrem junto às células que representam as impedâncias associadas às vias (ruas ou estradas), portanto, são consequência das imagens geradas por estas ferramentas específicas, que converteram de maneira diferente o tema vectorial de entrada.

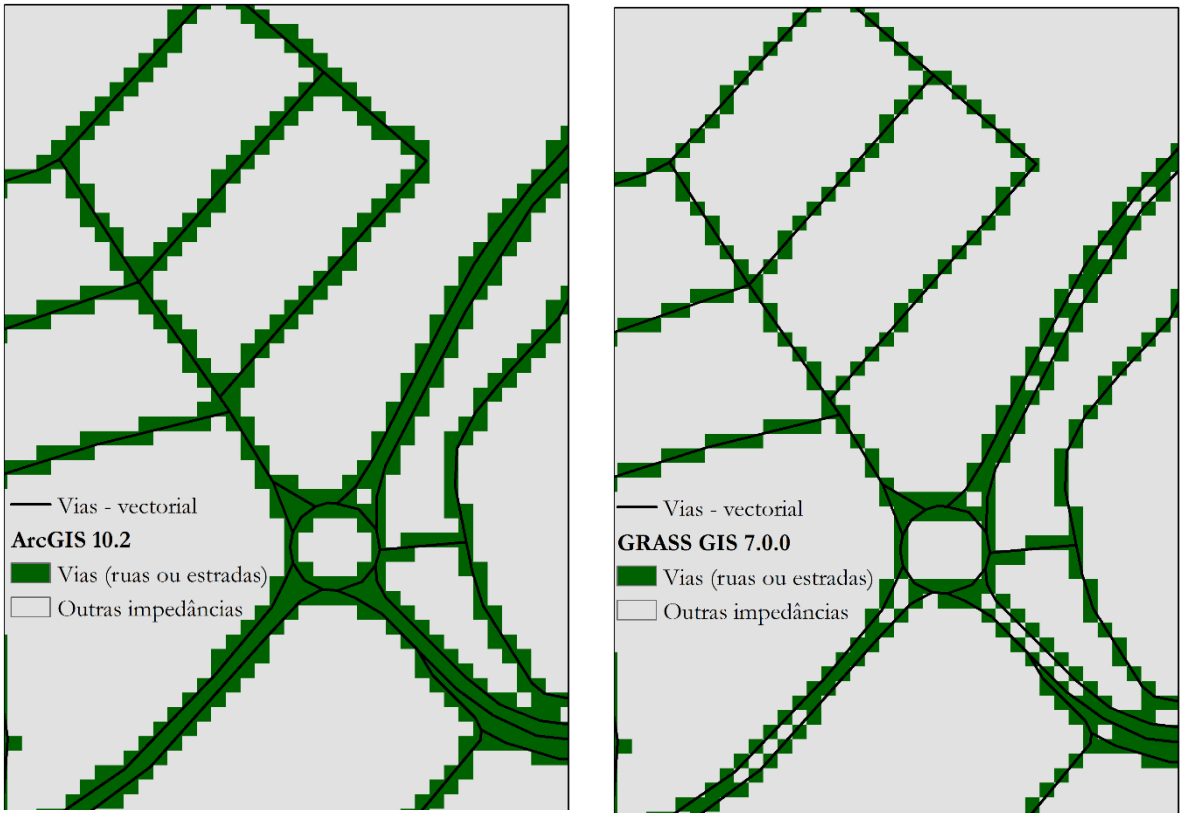


Figura 71 - Distorções presentes nos *outputs* das ferramentas *Polyline to Raster* (ArcGIS) e *v.to.raster* (GRASS GIS).

[218] **Ver Anexo A, p. A11.**

A procura por uma explicação para o sucedido levou-nos a analisar os parâmetros de entrada das *tools* de ambas as aplicações, tarefa que nos permitiu constatar que a *v.to.raster,* ao contrário da *Polyline To Raster* do ArcGIS, dá ao utilizador a possibilidade de escolher a densidade das linhas, isto é, se o operador optar por definir como parâmetro de entrada a *flag –d*, o algoritmo vai criar um *raster* em que todas as células em que a linha toca assumirão um valor diferente de *null* ou *nodata*; se esta *flag* não for incluída, no *raster* de saída só terão valor as células necessárias para garantir a conectividade topológica da linha. A utilização desta *flag* faz com que o resultado seja idêntico tanto no GRASS como no ArcGIS, portanto, a sua não utilização na realização desta tarefa, explica as diferenças nos *outputs* em comparação.

O facto acima aludido mostra quão importante é o estudo e o conhecimento dos parâmetros de configuração de uma ferramenta no software SIG, e como, em consequência de desatenções, desconhecimentos ou imprecisões técnicas do próprio algoritmo podem resultar em produções cartográficas com divergências mais ou menos significativas e importantes. Devido ao facto de ser o nosso primeiro contacto com a maior parte das ferramentas do GRASS GIS, não nos apercebemos da existência desta *flag* e, por esse facto, produzimos um resultado que nos causou estranheza, embora julguemos por correcta a interpretação que apresentamos sobre a ocorrência.

Voltando à semelhança arquitectural entre os Programas 40 e 41, nem sempre foi possível assegurar a utilização de uma cadeia de processos exactamente idêntica, ao contrário do que se verifica neste exercício, em particular, no qual o trabalho se tornou mais simples, visto que não houve necessidade de se procurar desvios à cadeia de geoprocessos inicialmente delineada. Isto não seria possível com outros SL/CA SIG, pois nenhum deles tem, no momento, uma biblioteca de ferramentas de geoprocessamento tão rica e completa como a do GRASS GIS, claramente equiparável à do ArcGIS.

As únicas divergências entre estas rotinas, dignas de nota de destaque, estão no modo como se acede aos dados alfanuméricos associados às *features* e às células de um *raster*. No primeiro caso, como se disse anteriormente, o GRASS armazena os dados geográficos e os alfanuméricos em espaços diferentes da memória com o intuito de optimizar a eficiência em análises estritamente espaciais, nas quais as informações da tabela de atributos não são relevantes. Assim, quando o operador pretende editar estes dados no GRASS, tem, *a priori*, de estabelecer uma conexão à tabela de atributos, usando o módulo *sqlite3*. Esta é, porventura, uma das adaptações mais enfadonhas que o operador que migra para o GRASS GIS tem de fazer.

Em segundo lugar, no GRASS, o acesso a dados relevantes associados a um *raster* faz-se de um modo distinto quando comparado ao ArcGIS. Por exemplo, a ferramenta *r.cross* (análoga à *combine*

do ArcGIS 10.2) agrega duas imagens matriciais numa só; os valores resultantes dizem respeito a categorias que representam todas as combinações possíveis entre os valores de entrada. Enquanto no ArcGIS esta informação fica disponível na tabela de atributos, no GRASS temos de a requisitar com a ferramenta *r.report* e manipulá-la a partir de um ficheiro de texto.

Na 2ª Tarefa, o panorama inverte-se, destacando-se a expressiva vantagem do GRASS GIS na execução da ferramenta para cálculo do custo acumulado, especialmente com o aumento do volume de dados (Figura 72) [219]. A ferramenta *r.cost* revela-se extremamente eficiente, trabalhando aproximadamente 67 milhões de células em menos de dois minutos (apenas 8,7% do tempo gasto pelo ArcGIS). O impacto do volume de dados é praticamente inexpressivo.

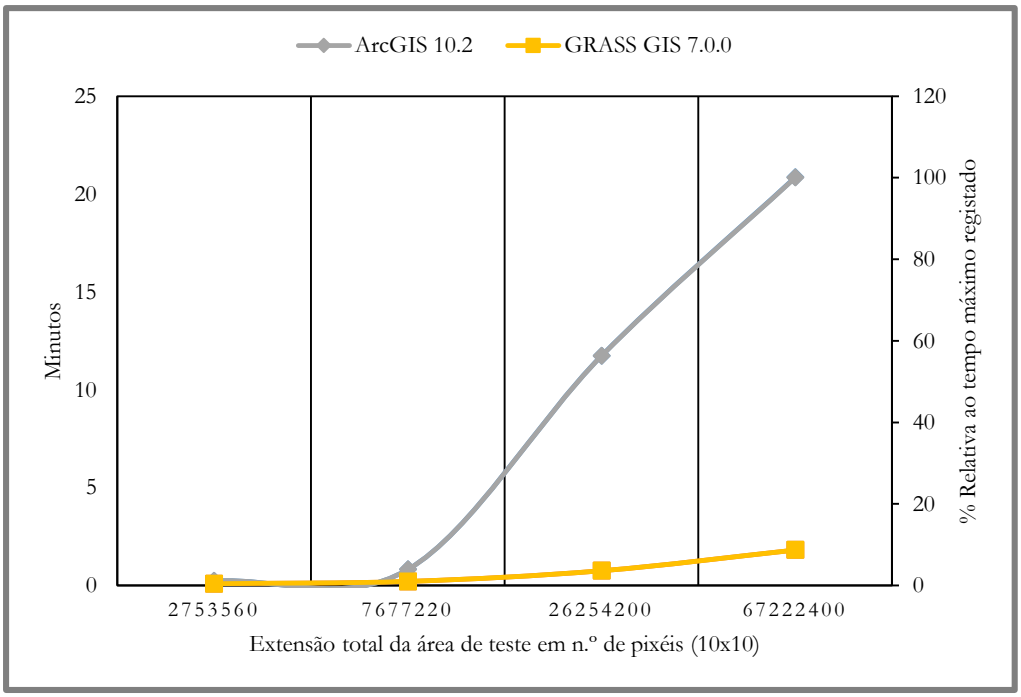


Figura 72 - Tempo consumido na execução da tarefa de criação de um cartograma de isócronas (custo acumulado), tendo em conta o aumento do volume dos dados de entrada.

Pelo contrário, o ArcGIS atrasa-se substancialmente com o aumento do volume de dados, sobretudo a partir das dezenas de milhões de células, necessitando de 21 minutos para concluir a tarefa para a área de teste 4.

219

	2 753 560 Células	7 677 220 Células	26 254 000 Células	67 222 400 Células
ArcGIS 10.2	00:00:14	00:00:49	00:11:44	00:20:51
GRASS GIS 7.0.0	00:00:05	00:00:12	00:00:45	00:01:49

Sem precisarmos de analisar o código-fonte destas ferramentas, percebemos claramente que se trata de arquitecturas distintas entre as ferramentas das duas aplicações SIG, pois não é simples encontrar outra explicação razoável para que se percebam as diferenças entre os tempos consumidos pelos dois programas. Estruturas distintas que não se traduzem em resultados cartográficos[220] com divergências significativas, quer visuais (Figura 73 e Figura 74), quer estatísticas (correlação de 99,9%). Em todo o caso, a nível decimal, maioritariamente a partir da segunda casa, verificámos também a existência de pixéis com pequenas distorções.

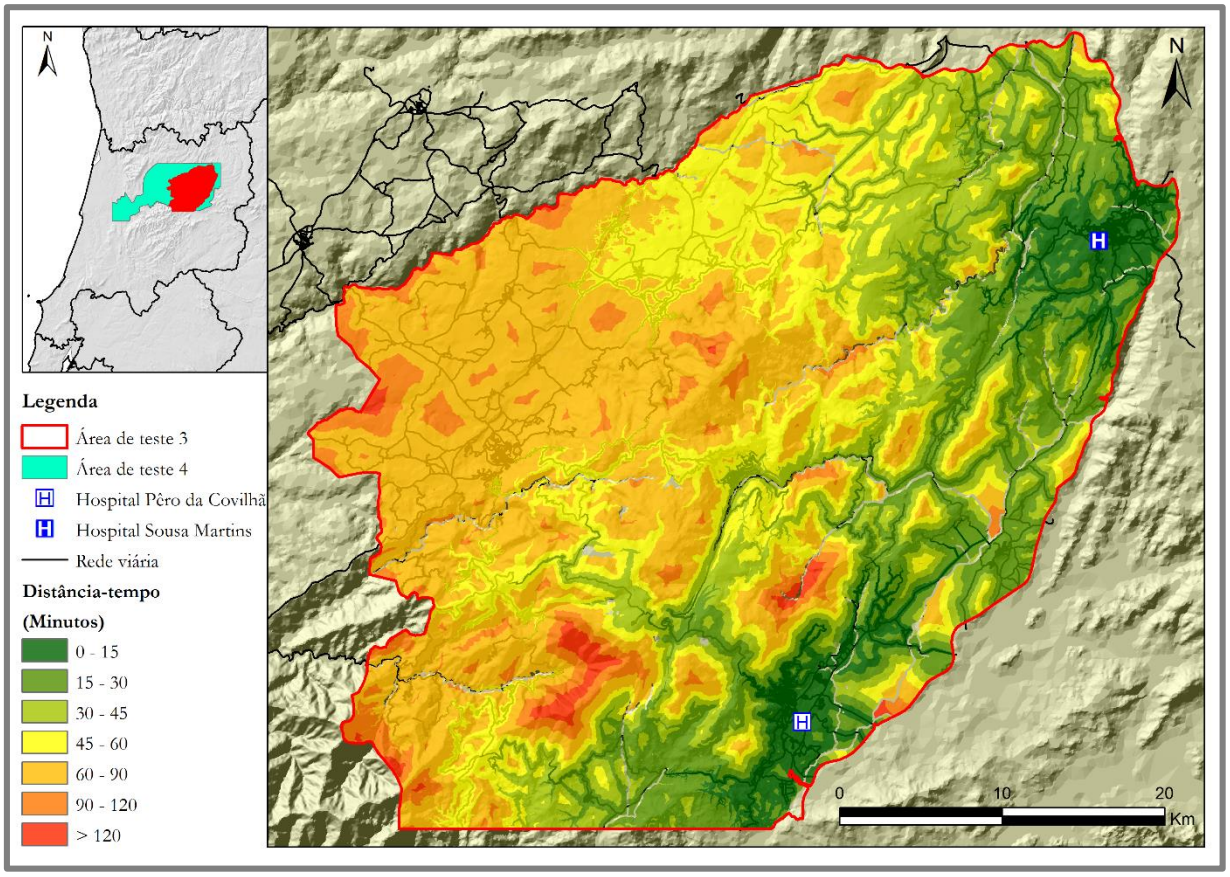


Figura 73 - Distribuição espacial de células isócronas que expressa a acessibilidade física aos hospitais considerados – resultado do Programa 42 executado sobre ArcGIS 10.2.

[220] Estes foram gerados a partir da superfície de custo construída no ArcGIS, pois, uma vez que existem pequenas diferenças nas superfícies de custo que constam da Figura 70, os resultados do *cost distance* e *r.cost* seriam necessariamente diferentes, algo que não pretendíamos.

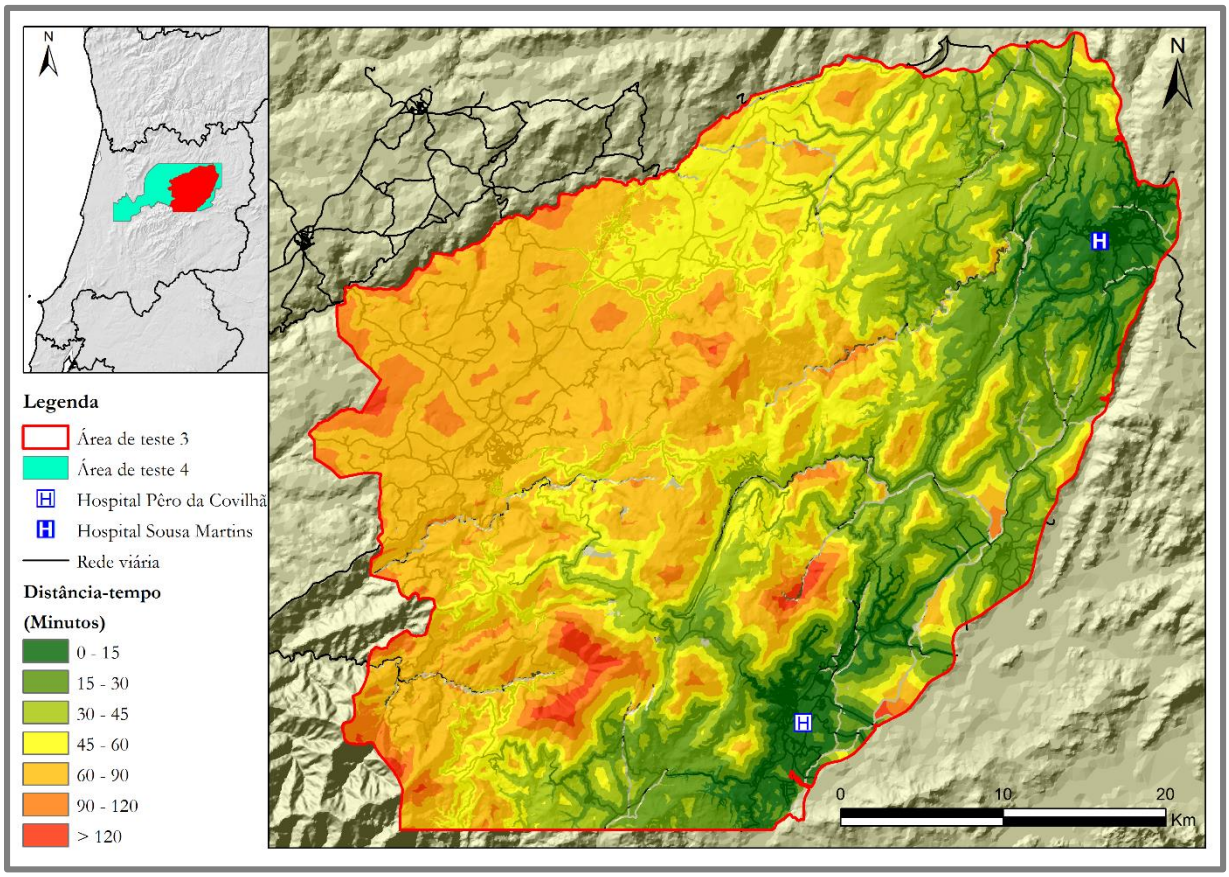


Figura 74 - Distribuição espacial de células isócronas que expressa a acessibilidade física aos hospitais considerados – resultado do Programa 43 executado sobre GRASS GIS 7.0.0.

Por fim, a execução da 3ª Tarefa[221], permite verificar que voltamos a ter nova inversão da evolução dos tempos registados com a variação do volume de dados, novamente com vantagem para o ArcGIS.

Segundo a Figura 75[222], o ArcGIS apresenta uma variação de tempo idêntica à da Figura 72, ainda que seja interessante constatar que, para a área de teste 3, este *software* tenha demorado menos tempo (menos 4 minutos, aproximadamente) do que na tarefa anterior, apesar do procedimento do Programa 44[223] incluir muitas mais tarefas de geoprocessamento. Isso deve-se provavelmente ao facto de neste último exercício se terem usado mais pontos de origem (quartéis de bombeiros)

[221] Em que se relaciona o cartograma das Figuras 73 e 74 com a impedância relativa a deslocações a partir de locais onde estão alojados equipamentos que garantem a resposta em situações de emergência, para posterior associação com a distribuição da população.

[222]

	2 753 560 Células	7 677 220 Células	26 254 000 Células	67 222 400 Células
ArcGIS 10.2	00:00:24	00:01:30	00:07:51	00:22:37
GRASS GIS 7.0.0	00:02:45	00:29:55	Maior que 06:00:00	Maior que 06:00:00

[223] **Ver Anexo A, p. A74.**

do que na anterior, em que se usaram apenas os hospitais contidos nesta área de teste (o da Guarda e o da Covilhã). Havendo mais pontos de origem, logicamente, torna-se menos exigente, para a *tool cost distance,* encontrar os caminhos mais curtos, visto que existem menos percursos alternativos.

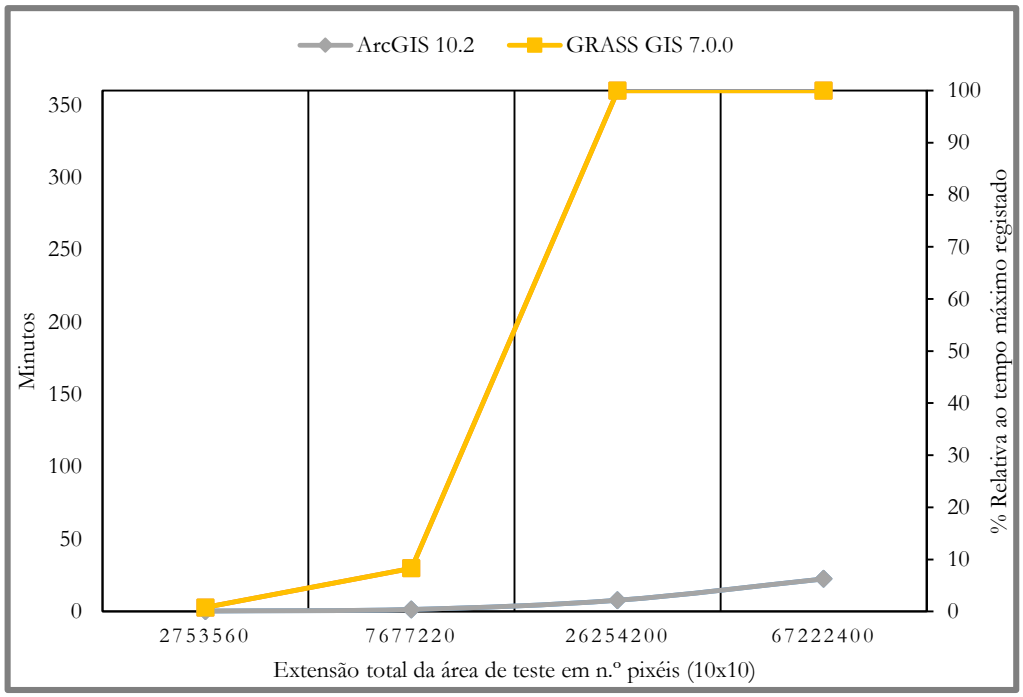


Figura 75 - Tempo consumido na execução da tarefa de criação de um cartograma de isócronas (custo acumulado) que representa a capacidade de resposta das equipas de emergência, tendo em conta o aumento do volume dos dados de entrada.

O GRASS GIS revelou mais dificuldade em resolver as tarefas propostas como se verifica logo no teste para a área 2 (delta de 27 minutos relativamente ao tempo despendido pelo ArcGIS). No que concerne às áreas de teste 3 e 4, em seis horas, o primeiro não foi capaz de completar o procedimento. Ao correr o Programa 45[224] pudemos acompanhar (monitorizar) em tempo real a janela MS-DOS que abre juntamente com o GRASS e que discrimina o estado de evolução da sequência ordenada de geoprocessos. Com esta monitorização constatámos um aumento da lentidão da resposta quando o *software* iniciou as tarefas de intersecção entre as várias classes de acessibilidade e os pontos que representam a população residente.

A ferramenta *v.select* é incapaz de devolver, iterativamente, a intersecção dos pontos (indivíduos) com os diferentes intervalos de distância-tempo, o que parece ir ao encontro dos resultados obtidos no Exercício 1 (Capítulo II da 2ª Parte), no qual a intersecção entre *features* era fundamental para a

[224] **Ver Anexo A, p. A78.**

obtenção do *output* final. Se, anteriormente, não percebemos (porque a área de estudo detinha menor extensão de área), parece-nos, agora, claro que a menor performance e desempenho do GRASS se podem justificar com a fraca eficiência deste geoprocesso, em específico.

Por fim, fazemos referencia à Tabela XLII (p. 238), na qual observámos que a sintaxe do código que construímos para o Programa 45 era mais complexo que o Programa 44. Esse apontamento merece-nos uma nota relacionada com uma limitação funcional associada à ferramenta *v.random*. No Programa 44, criam-se pontos aleatórios para representar os indivíduos que residem em cada célula da grelha, recorrendo à informação sobre os quantitativos populacionais que consta da tabela de atributos. Isto é possível porque a ferramenta *create random points* do ArcGIS 10.2 permite criar um número de pontos aleatórios diferente para cada *feature,* número esse que será igual a um valor que determinada *feature* apresenta num campo numérico, no nosso caso, o campo *Pop_tot*, no qual estão registados os totais de população residente verificados em cada *feature*/célula. Ao contrário desta, a *v.random* não possui a opção de criar, numa *feature* do vector de entrada, tantos pontos quantos os que constam num atributo/campo. O parâmetro de entrada que pede a definição de um nome de uma coluna diz respeito à escrita dos valores da coordenada Z, no caso de o utilizador pretender criar pontos aleatórios com altitude associada, nada tendo a ver com o parâmetro que nos importa nesta tarefa.

Assim, foi necessário construir uma ferramenta que resolvesse os problemas criados por esta limitação, e é por detrás desta necessidade que se encontra a diferença no indicador de complexidade da Tabela XLII. Este é um exemplo de como os conhecimentos em programação podem ser cruciais para resolver limitações e condicionalismos existentes nas ferramentas da biblioteca de algoritmos, se bem que, neste caso concreto, poderíamos ter também utilizado a *qgis:random points inside polygons variable*, ferramenta do QGIS que responde a esta necessidade.

Capítulo IV – Reflexões em torno das inconformidades estatísticas e/ou cartográficas resultantes da aplicação de ferramentas análogas

Como antes se referiu, a dado momento, na implementação de uma nova solução informática, torna-se legítimo que os administradores se questionem sobre se a alternativa que equacionam é eficaz matematicamente, logo, para além da comparação de performance e desempenho, importa proceder também a uma rigorosa avaliação de possíveis inconformidades matemáticas e estatísticas existentes nos *outputs*.

Durante o desenvolvimento desta dissertação, apercebemo-nos da existência de algumas discrepâncias decorrentes da execução de algoritmos comummente aplicados no âmbito da detecção remota e da modelação espacial. Assim, reunimos um conjunto de pequenos exercícios que pretendem demonstrar que, por vezes, apesar dos dados de entrada serem exactamente iguais e as ferramentas serem consideradas análogas (ou seja, implementam exactamente o mesmo método), os resultados gerados por diferentes *software* podem não ser absolutamente iguais.

Atente-se que em nenhum momento nos propomos avaliar a qualidade dos algoritmos testados, dizendo que um implementa um determinado método melhor que o outro, com estes pequenos ensaios, pretende-se, antes, convidar o leitor a reflectir acerca de problemas em torno da autenticidade de determinados mitos que referem que os resultados gerados por ferramentas análogas não é o mesmo. Portanto, as medidas usadas na comparação dos vários *outputs* que apresentamos servem apenas para identificar e avaliar a dimensão de eventuais diferenças entre estes últimos, e nunca para formular considerações sobre a sua exactidão temática e posicional, para as quais se teria de usar outras medidas, nomeadamente o erro médio quadrático.

Sobre os resultados, não faremos quaisquer juízos sobre o maior ou menor significado de quaisquer diferenças registadas, deixando a interpretação da sua magnitude à interpretação de cada um. Isto porque julgamos que tal interpretação é, de alguma forma, um exercício pessoal. Por um lado, um técnico ou administrador, ao migrar do *software* A para o B, pode achar que, a mais ligeira variação dos resultados gerados pelo segundo, origina complicações no normal *workflow* do organismo. Por

outro lado, há quem possa argumentar que, desde que a correlação seja forte e positiva, quaisquer dissemelhanças entre os *outputs* de algoritmos análogos são pouco significativas.

Apresentadas estas observações preambulares elencamos, de modo sumário, os ensaios realizados:

1. **Fusão de imagens multiespectrais tendo em vista a melhoria da sua resolução espacial**
   i. Implementar, em diferentes *software* SIG *desktop*, duas estratégias que visam a optimização da resolução espacial das bandas relativas ao visível de uma imagem multiespectral, através da sua fusão com a banda pancromática.
   ii. Comparar visualmente e estatisticamente os resultados obtidos.
2. **Exemplos aplicados à análise morfométrica do relevo**
   i. Executar, em diferentes *software* SIG *desktop*, o interpolador *Inverve Distance Weighted* (IDW) com o fim de criar um MDT.
   ii. Derivar, em diferentes *software* SIG *desktop*, a exposição das vertentes a partir de um MDT.
   iii. Comparar visualmente e estatisticamente os resultados obtidos.

I. Fusão de imagens multiespectrais tendo em vista a melhoria da sua resolução espacial

"*Os olhos humanos são sensores que captam a luz reflectida pelos objectos, por isso, são o sistema de detecção remota mais vulgar.*"

É com frequência que se encontram algoritmos/funcionalidades para processamento de dados adquiridos no âmbito da Detecção Remota (DR) em *software* SIG *desktop* (Steiniger & Hunter, 2013). Apesar de as noções e aplicações da DR encontrarem-se à parte do conceito de SIG, com eles incorporando um conceito mais amplo – o de TIG (cfr. Figura 8, p. 47), estes termos assumem uma estrita relação, na medida em que os dados obtidos remotamente são cruciais para a modelação espacial em ambiente SIG.

A DR é a ciência e técnica para a obtenção de informação sobre um objecto, área ou fenómeno através da análise de dados adquiridos por um dispositivo, sem que ele esteja em contacto directo com o primeiro (Konecny, 2003). De modo geral, a DR faz-se com sensores que detectam a energia emitida ou reflectida pelos objectos, sendo capazes de registar dados espectrais em várias zonas do espectro electromagnético (bandas ou canais onde existem janelas atmosféricas[225]), sendo esses valores posteriormente convertidos em números digitais que traduzem a radiação reflectida – nível radiométrico, registando-os numa matriz regular (imagem).

Entre as várias imagens produzidas no âmbito da DR, destacam-se as imagens multiespectrais, que são constituídas por sensores passivos (ou seja, não têm a sua própria fonte de radiação) e que por norma têm uma boa resolução espectral, em detrimento da resolução espacial (tamanho da entidade mais pequena que pode ser detectada)[226].

[225] As janelas atmosféricas são bandas espectrais com uma elevada transmitância atmosférica (zonas do espectro onde os efeitos da absorção são fracos), através dos quais a radiação passa livremente do espaço para a Terra e vice-versa.
[226] A resolução espacial varia com o comprimento de onda, uma vez que, segundo as leis da física, a energia de um corpo é inversamente proporcional ao seu comprimento de onda, assim, é natural que as bandas que operam num maior comprimento de onda tenham de receber radiação electromagnética de uma maior extensão de área, isto para que a quantidade de energia que chega ao sensor seja detectada, e claro, quando maior for a área a ser captada, menor será a resolução e o detalhe da superfície.

Com o intuito de melhorar a resolução espacial de uma imagem sem degradar as suas propriedades radiométricas, encontrámos na bibliografia da especialidade, referência a vários autores que propõem diferentes métodos que visam a fusão das bandas multiespectrais com a banda pancromática da imagem, juntando-se a qualidade espectral das primeiras com a maior resolução espacial da segunda.

Os pacotes de *software* mais utilizados disponibilizam ferramentas (análogas) que permitem a resolução deste tipo de tarefa, mas será que os resultados são idênticos? Para poder dar resposta a esta questão, desenvolvemos dois breves exercícios que incrementam duas metodologias frequentemente aplicadas quando o objectivo é a melhoria da resolução espacial de uma imagem multiespectral, a saber:

i. Transformação IHS (*Intensity, Hue, Saturation*);

ii. E transformação segundo o Algoritmo de Brovey.

1.1. Definição da área de teste

Antecedendo a execução destes dois exercícios, procedemos à delimitação de uma área de teste (Figura 76), para a qual preparamos um extracto de uma imagem Landsat 8, descarregada gratuitamente a partir da plataforma *Earth Explorer* (http://earthexplorer.usgs.gov/, acedido em 16 Dezembro, 2015). A escolha desta área não se deve a nenhuma outra razão que não a de obtenção de uma área de teste que permitisse a comparação entre os resultados dos métodos aplicados nos exercícios.

Neles consideramos apenas as três bandas espectrais do visível (banda do azul, do verde e do vermelho) com uma resolução de 30 metros, e a banda pancromática com uma resolução de 15 metros.

Estas bandas passaram por um momento de pré-processamento, em que ajustamos a sua resolução radiométrica – passagem de 16 para 8 *bits*, corte da cauda do histograma em 5%, e alteração da resolução espacial das bandas espectrais para 15 metros (a mesma da banda pancromática). À semelhança dos exercícios anteriores, a preparação dos dados de entrada (pré-processamento) foi executada apenas num *software*, o PCI Geomatica. Naturalmente, os dados resultantes destas tarefas preliminares foram usados em todos os pequenos ensaios de fusão de imagens como elementos de entrada, tornando válida a comparação estatística entre os *outputs*.

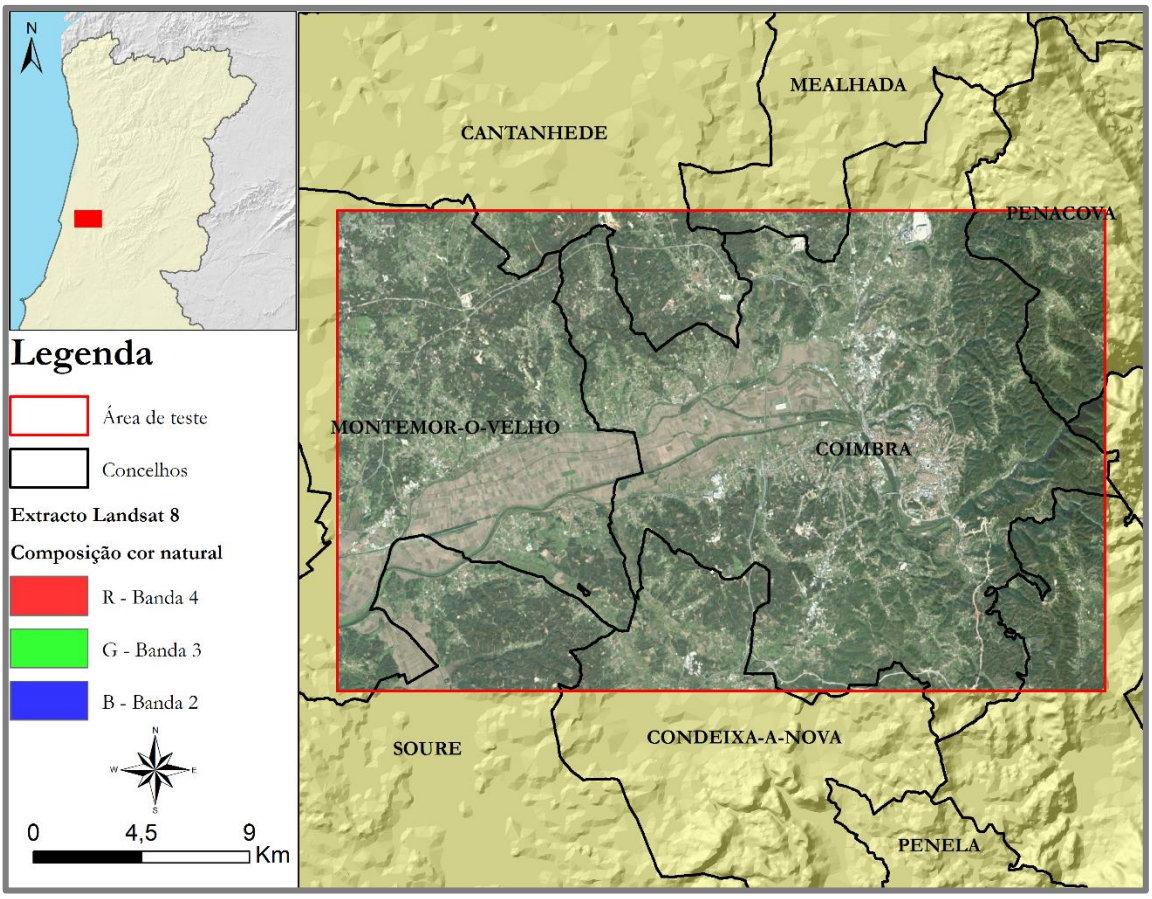


Figura 76 - Localização do extracto da imagem multiespectral Landsat 8 utilizada na execução dos exercícios que visam a melhoria da sua resolução espacial.

1.2. Transformação IHS

Em termos operacionais, a transformação IHS é concretizada segundo o seguinte conjunto de operações:

i. Conversão das bandas cujo comprimento de onda se encontra na zona do espectro electromagnético que diz respeito ao visível (RGB – *red, green, blue*) em novas imagens que representam a intensidade, brilho e saturação deste conjunto (RGB > IHS);

ii. Nova conversão das componentes IHS para RGB, substituindo a componente intensidade pela banda pancromática.

Este procedimento foi executado em três *software*: PCI Geomatica (cujo procedimento é apresentado na Figura 77); Sextante integrado no gvSIG 2.2; e GRASS GIS 7.0.0. Neste último,

incrementaram-se dois modos de o fazer (Programa 46[227] e 47[228]) - no Programa 46, executamos as ferramentas *i.rgb.ihs* e a *i.ihs.rgb* para realizar as conversões e alterações explicadas (exactamente como se fez no PCI Geomatica – Figura 77); no Programa 47, recorre-se somente à ferramenta *i.pansharpen*, tendo-se definido, como método a utilizar, a opção IHS. No gvSIG, usaram-se as ferramentas *HIS -> RGB* e *RGB -> HIS* do *Image Processing* da *toolbox* Sextante, em jeito igual ao apresentado na Figura 77 e Programa 47.

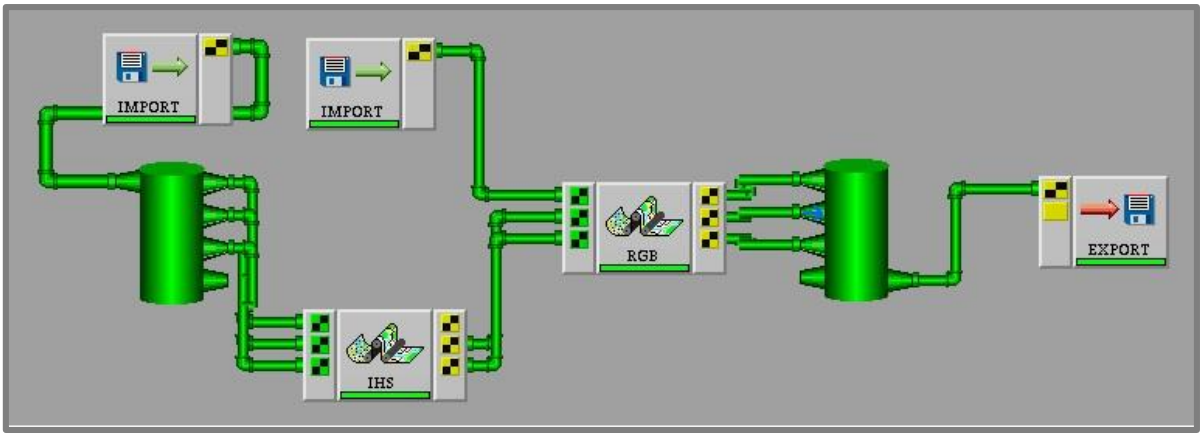
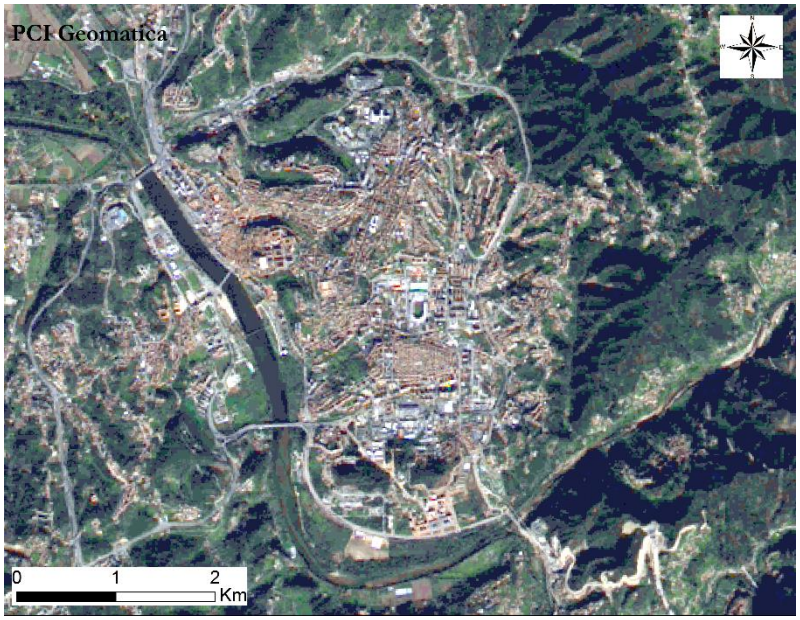
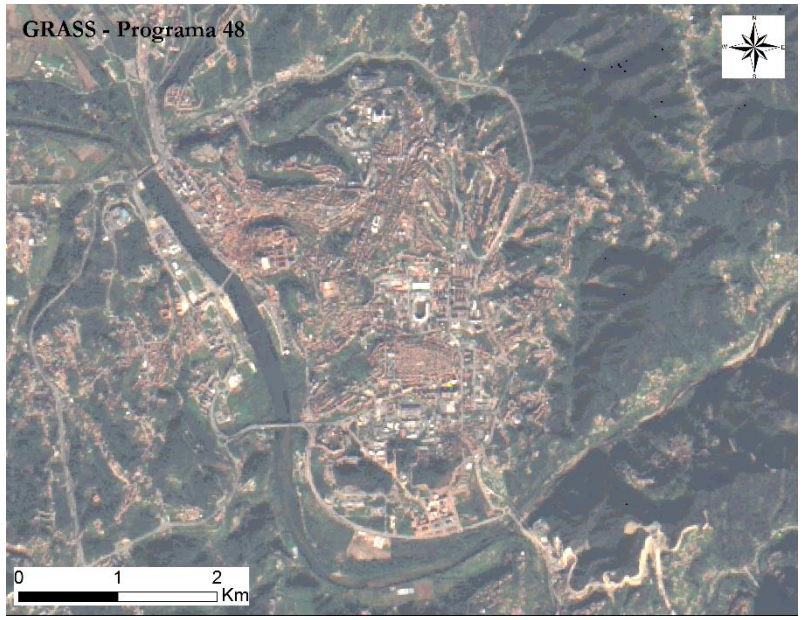


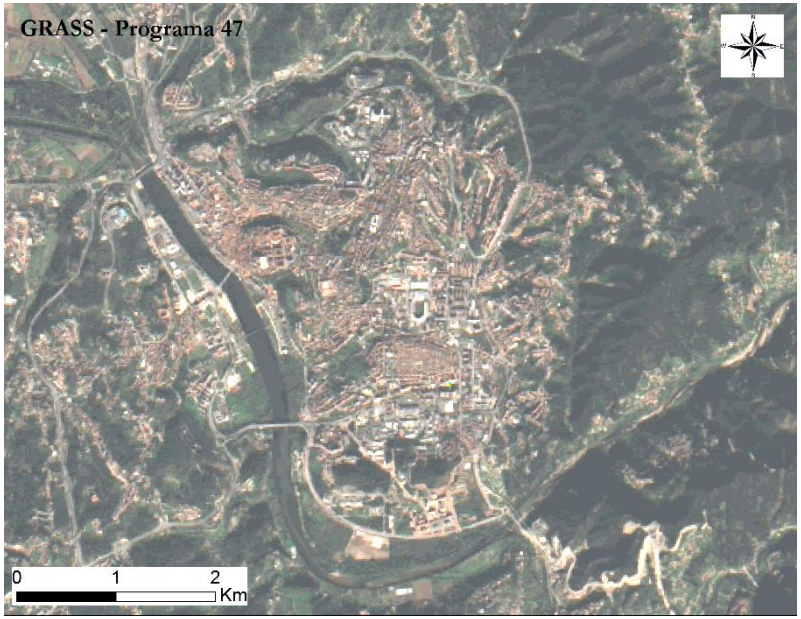
Figura 77 - Modelo que executa a Transformação IHS no PCI Geomatica.

A Figura 78 apresenta os resultados obtidos e deixa notar que, em termos visuais, existem perceptíveis disparidades nas tonalidades da cor. A nível estatístico, a correlação linear entre as várias imagens envolvidas neste exercício (Tabela XLIII) comprova que os resultados não são absolutamente iguais, do mesmo modo que as distorções por eles causadas por eles nos níveis radiométricos das bandas originais, também não são iguais.

[227] **Ver Anexo A, p. A81.**
[228] **Ver Anexo A, p. A82.**







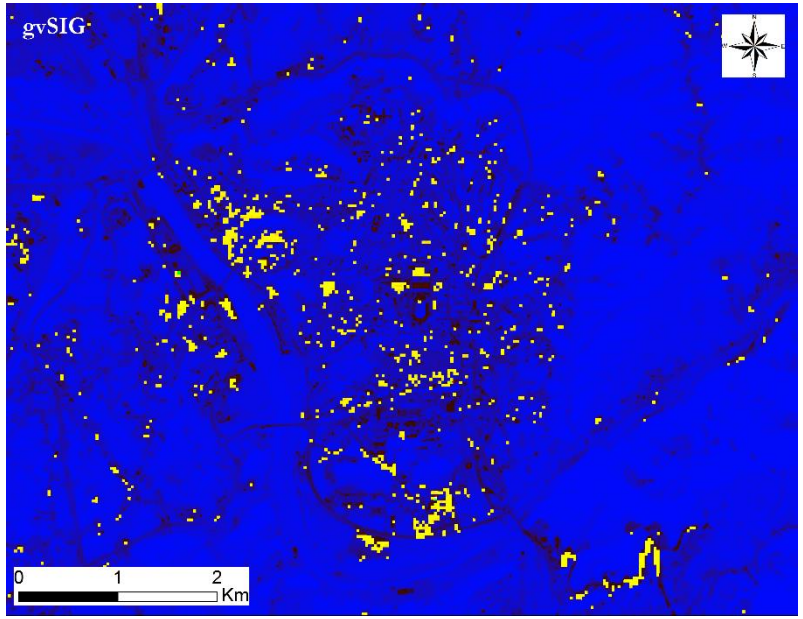


Figura 78 - Imagens que resultaram da transformação IHS nos diferentes *software* em confronto.

Tabela XLIII - Matriz de correlação entre as várias bandas das imagens resultantes da aplicação do método de transformação/fusão IHS em diferentes *software*

		Original			PCI			GRASS (Prog. 46)			GRASS (Prog. 47)		
		R	G	B	R	G	B	R	G	B	R	G	B
Original	R	1											
	G		1										
	B			1									
PCI	R	0,90			1								
	G		0,91			1							
	B			0,96			1						
GRASS (Prog. 46)	R	0,96			0,96			1					
	G		0,95			0,97			1				
	B			0,95			0,98			1			
GRASS (Prog. 47)	R	0,96			0,93			0,98			1		
	G		0,95			0,94			0,97			1	
	B			0,94			0,96			0,97			1
gvSIG	R	0,43			0,45			0,44			0,42		
	G		0,15			0,17			0,17			0,13	
	B			0,73			0,71			0,67			0,70

Não podemos deixar de fazer referência ao completo desajuste do resultado gerado pelo gvSIG/Sextante relativamente àquilo que era expectável, de tal modo que, a nível visual, não é possível distinguir os vários objectos que compõem a malha urbana da cidade de Coimbra. Esta desvirtuação visual é confirmada pelos resultados estatísticos, que exprimem claramente que o maior desajustamento refere-se à banda verde, se bem que, quando comparada com as restantes, a correlação é também mais baixa nas outras bandas, nomeadamente na vermelha. Face ao exposto, estamos em condições de afirmar que os algoritmos da *toolbox* Sextante utilizados apresentam um funcionamento que deve ser alvo de revisão e melhoria urgentes.

Por fim, não podemos deixar de destacar a curiosidade que consiste no facto de, no mesmo *software* (GRASS GIS), existirem duas ferramentas que se destinam ao mesmo fim, mas que geram resultados com algumas dissemelhanças.

Numa reflexão pouco exaustiva sobre estes resultados, admitimos, ainda que com algum cepticismo, que estes indicadores se ficam a dever à existência ou inexistência de contingências relacionadas com ajustamentos específicos do algoritmo a produtos de imagem de determinado sensor. Por exemplo, a *i.pansharpen* do GRASS GIS possui uma *flag* que visa dar um tratamento específico ao canal azul se se tratar de uma imagem Landsat. Esta opção concorre com outra justificação, que tem a ver necessariamente com a existência de assimetrias lógicas e processuais na gramática do código-fonte dos vários algoritmos, dentro e fora do mesmo *software*.

I.3. Transformação segundo o método de Brovey

Tanto o ArcGIS como o GRASS GIS dispõem de uma ferramenta que aplica o método de Brovey, uma técnica que, à semelhança da IHS, procura optimizar a resolução espacial das bandas espectrais através da sua fusão com a banda pancromática. Assim sendo, usando mais uma vez o extracto de uma imagem Landsat apresentado na Figura 76, comparamos os *outputs* resultantes das seguintes ferramentas:

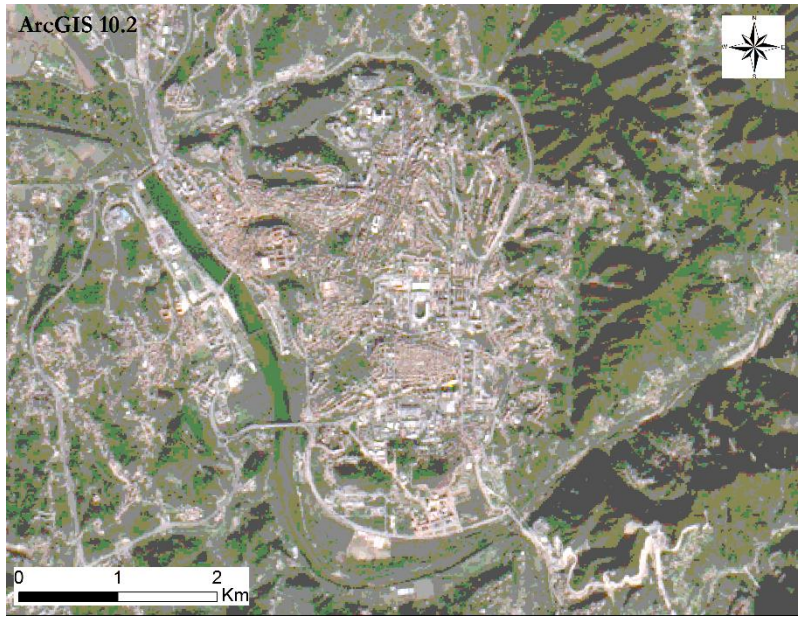
i. ArcGIS 10.2 – *Arctoolbox* > *Data Management Tools* > *Raster* > *Raster Processing* > *Create Pan-sharpened Raster Dataset*;

ii. GRASS GIS 7.0.0 – *i.pansharpen*, aplicando, agora, a opção Brovey.

À semelhança daquilo que se verificou na transformação IHS, denotam-se algumas distorções entre os dois *outputs*, em concreto, na Figura 79 verificam-se distorções na cor entre as duas imagens produzidas. Quanto aos coeficientes de correlação (Tabela XLIV), as dicotomias entre os dois resultados são mais salientes do que as registadas na Tabela XLIII. O maior desajustamento da banda azul gerada pelo GRASS relativamente à mesma banda da imagem original deve-se, provavelmente, à não utilização da *flag* a que antes nos referimos.

Tabela XLIV - Matriz de correlação entre as várias bandas das imagens resultantes da aplicação do método de transformação/fusão de Brovey, no ArcGIS 10.2 e GRASS GIS 7.0.0

		Original			ArcGIS 10.2		
		R	G	B	R	G	B
Original	R	1					
	G		1				
	B			1			
ArcGIS 10.2	R	0,88			1		
	G		0,80			1	
	B			0,75			1
GRASS GIS 7.0.0	R	0,91			0,95		
	G		0,86			0,94	
	B			0,65			0,91



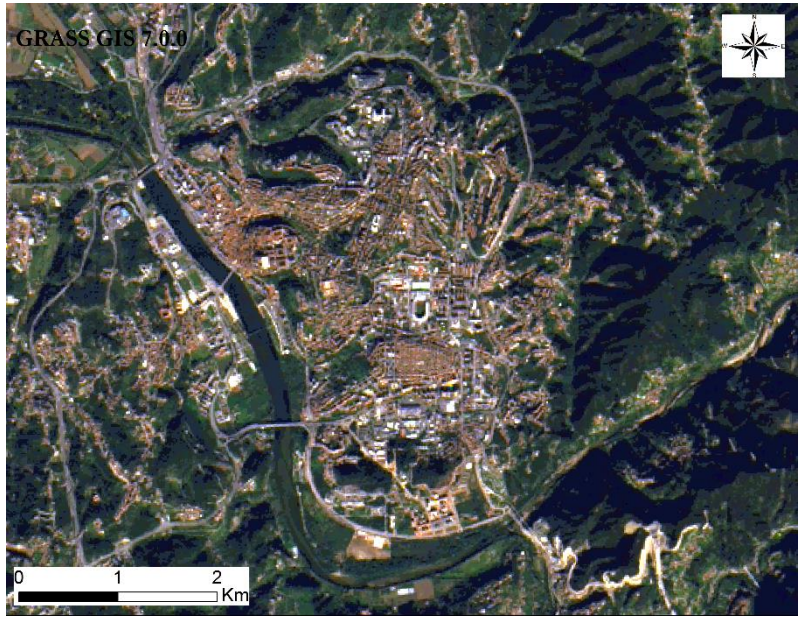


Figura 79 - Extracto de uma imagem Landsat 8 original e imagens que resultaram da aplicação do método de Brovey no ArcGIS 10.2 e GRASS GIS 7.0.0.

2. Exemplos aplicados à análise morfométrica do relevo

> "*As notícias reconfortantes são que o método de Horn e o algoritmo de Zevenbergen e Throne são usados por grande parte dos software SIG proprietários de referência, o que indica que os seus produtores concordam que estes são os melhores algoritmos.*"
>
> Peter Burrough & Rachael McDonnell (2000:192)[229]

Imagine-se um cenário em que uma organização pretende saber quais são as melhores áreas para a instalação de paneis fotovoltaicos. Entre as variáveis que integram o modelo que dá solução ao problema está, por motivos óbvios, a exposição das vertentes. Em termos operacionais, a derivação de um mapa de exposições é relativamente simples, isto porque o utilizador tem à sua disposição algoritmos que incrementam vários métodos que visam a concepção de um MDT e a derivação, a partir dele, da carta de exposições.

Numa primeira fase, suponhamos que optamos pela utilização do interpolador *Inverse Distance Weighted* (IDW) para construção do MDT. Esta técnica está muito bem documentada em Burrough & McDonnell (2000), mas será que está implementada exactamente da mesma maneira nos vários *software* SIG *desktop*?

Em busca de possíveis respostas, considerando a área de teste apresentada na Figura 80 (com 6 195 720 de pixéis - resolução de 10 metros), corremos os interpoladores IDW existentes em alguns dos *software* SIG *desktop* que disponibilizam esta ferramenta[230], a saber:

i. ArcGIS 10.2 – *Arctoolbox* > *Spatial Analyst Tools* > *Interpolation* >*IDW;*

ii. gvSIG 2.2 – Sextante > *Rasterization and interpolation* > *Inverse Distance Weighting (IDW);*

iii. GRASS GIS 7.0.0 – *v.surf.idw;*

---

[229] Tradução livre do original, "*The reassuring news is that Horn's method and the Zevenbergen and Thorne algorithm are used by several well-know commercial GIS, so that there is general agreement on the better algorithms.*" *In* BURROUGH, Peter & MCDONNELL, Rachael (2000) – **Principles of Geographical Information Systems**. Oxford University Press, 2ª Edição, Nova Iorque, 333 p.

[230] Como dados de entrada, foram utilizados os vértices das curvas de nível das folhas N.º 212, 213, 223 e 224 da Carta Militar de Portugal com escala de 1:25 000.

iv. SAGA GIS 2.1.2 – *Tool Libraries* > *Grid – Gridding* > *Inverse Distance Weighted;*

v. GDAL integrado no QGIS 2.10.1 – *Toolbox Processing* > *GDAL/OGR* > *Analysis* > *Interpolate (Inverse distance weighting).*

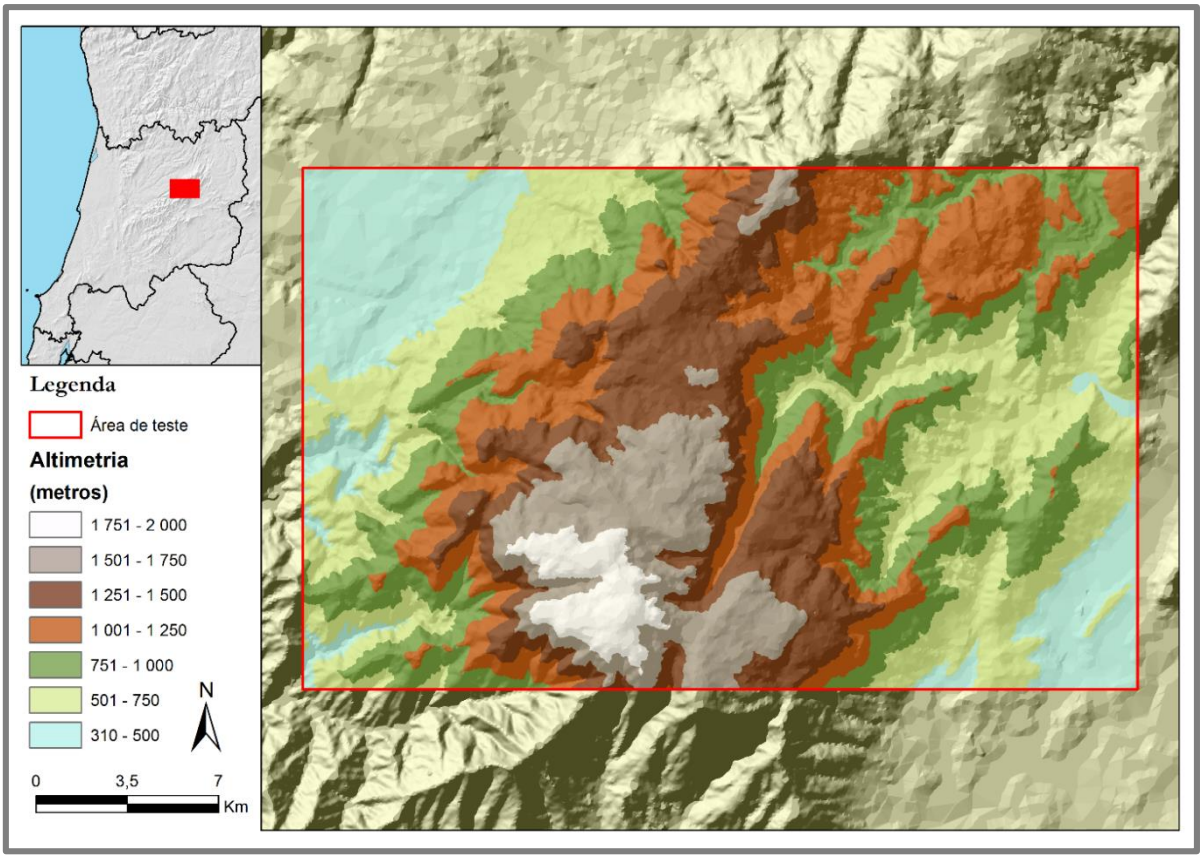


Figura 80 - Localização da área de teste onde, supostamente, se pretende instalar painéis fotovoltaicos.

No desenvolvimento deste exercício, atendendo à sensibilidade do algoritmo e de modo a tornar a comparação válida[231], certificamo-nos que todos os parâmetros de entrada eram idênticos, nomeadamente o expoente do inverso da distância (usaram-se 2), o número de pontos a considerar no cálculo da média pesada (usaram-se 12) e o raio/distância máxima de procura (definiram-se 100 metros de raio).

Em relação aos resultados obtidos, a matriz da Tabela XLV denota, apesar de tudo, uma grande semelhança entre as várias imagens – correlação forte positiva, com valores sempre muito próximos

[231] Uma vez que o interpolador IDW disponível no QGIS 2.10.1 não permite a definição destes parâmetros, e por sermos incapazes de analisar o seu código-fonte, programado em C++, este software foi excluído deste ensaio, visto que não é possível assegurar que os referidos parâmetros estão comparados da mesma forma, inviabilizando a legitimidade de comparação.

da unidade. Todavia, numa descida ao pormenor, a realidade lida a partir da Tabela XLVI, revela diferenças intrigantes, na medida em que, apenas uma ínfima parte das células apresentam valores exactamente idênticos, ainda que a esmagadora maioria das distorções não ultrapasse os 5 metros, com excepção feita para a biblioteca GDAL/OGR, pois, quando comparada com os outros, mais de metade dos pixéis da área de estudo apresentam diferenças superiores a 5 metros. Por seu turno, o SAGA GIS destaca-se dos demais por gerar o resultado mais idêntico ao do ArcGIS, registando-se diferenças até 3 metros em 116 pixéis (0,002%), e por ser aquele que também evidencia maior coeficiente de correlação.

Tabela XLV - Matriz de correlação entre as superfícies interpoladas segundo o método IDW em vários *software* SIG

	ArcGIS 10.2	gvSIG 2.2	GRASS 7.0.0	GDAL
ArcGIS 10.2	**1**			
gvSIG 2.2	0,992	**1**		
GRASS 7.0.0	0,975	0,982	**1**	
GDAL	0,984	0,985	0,974	**1**
SAGA 2.1.2	0,999	0,992	0,975	9,984

Tabela XLVI - Percentagem (do número de células) de diferenças entre as várias matrizes geradas no âmbito deste exercício

Percentagem de pixéis que apresentam diferenças				
	ArcGIS 10.2	**gvSIG 2.2**	**GRASS 7.0.0**	**GDAL**
ArcGIS 10.2	**0%**			
gvSIG 2.2	96,63%	**0%**		
GRASS 7.0.0	98,22%	99,05%	**0%**	
GDAL	92,93%	97,17%	98,74%	**0%**
SAGA 2.1.2	0,002%	96,63%	98,22%	92,93%
Percentagem de pixéis que apresentam diferenças superiores a 5 metros				
ArcGIS 10.2	**0%**			
gvSIG 2.2	2,91%	**0%**		
GRASS 7.0.0	4,96%	9,66%	**0%**	
GDAL	62,27%	61,91%	63,75%	**0%**
SAGA 2.1.2	0%	2,91%	4,96%	62,27%
Percentagem de pixéis que apresentam diferenças superiores a 50 metros				
ArcGIS 10.2	**0%**			
gvSIG 2.2	0,53%	**0%**		
GRASS 7.0.0	1,91%	1,38%	**0%**	
GDAL	2,29%	2,13%	3,35%	**0%**
SAGA 2.1.2	0%	0,53%	1,91%	2,29%

Como complemento, na Figura 81 apresentamos a distribuição das desigualdades pela área de teste e, para cada tipo de diferença (maior que 0; maior que 5 metros e maior que 50 metros),

expressamos o número de vezes que uma célula foi classificada como diferente mediante a comparação entre cada par de *outputs*[232]. Esta figura, sendo a expressão cartográfica dos dados alfanuméricos descritos nas duas tabelas previamente expostas, merece, da nossa parte, dois comentários:

i. O primeiro tem a ver com a distribuição do número de vezes que uma célula apresenta uma diferença superior a 5 metros. Como se verifica, grande parte dos pixéis assumem o valor 4, que resulta do facto destes serem diferentes na comparação entre a imagem gerada pela GDAL/OGR e todas as outras. Cremos que esta situação se fica a dever ao facto da *Interpolate (Inverse distance weighting)* ou *gdal_grid*, ao ser integrada no QGIS, perder a possibilidade de definição do tamanho da célula. Neste caso, a *tool* perde funcionalidades relativamente à utilização da biblioteca em contexto *standalone*, à semelhança do que acontece com a integração do *raster calculator* do GRASS GIS no QGIS. Isto fez com que o programa definisse automaticamente uma resolução maior do que aquela que pretendíamos, sendo provavelmente a causa para um maior número de células com diferenças superiores a 5 metros.

ii. Em segundo lugar, as diferenças superiores a 50 metros devem-se sobretudo a distintas condições e decisões subjacentes ao código-fonte da ferramenta *v.surf.idw* do GRASS GIS. Enquanto os interpoladores das outras distribuições atribuem o valor 0 ou *nodata* às áreas que não foram capazes de interpolar, a ferramenta *v.surf.idw* atribui-lhes o valor de altitude mais baixo em toda a extensão da área processada.

[232] Os cartogramas da Figura 81 resultam de um fluxo de trabalho como o que seguidamente se apresenta:
   i. Cálculo das diferenças (valor absoluto) entre todos pares de imagens possíveis;
   ii. Em cada uma das novas imagens representativas da diferença entre cada um dos pares, atribuiu-se o valor 1 às células que respeitavam as condições > 0; > 5 e > 50, gerando três novos ficheiros *raster*;
   iii. Para cada uma das condições, somaram-se todas as matrizes referentes a cada par de imagens comparadas, originando os três cartogramas representados na figura em questão.

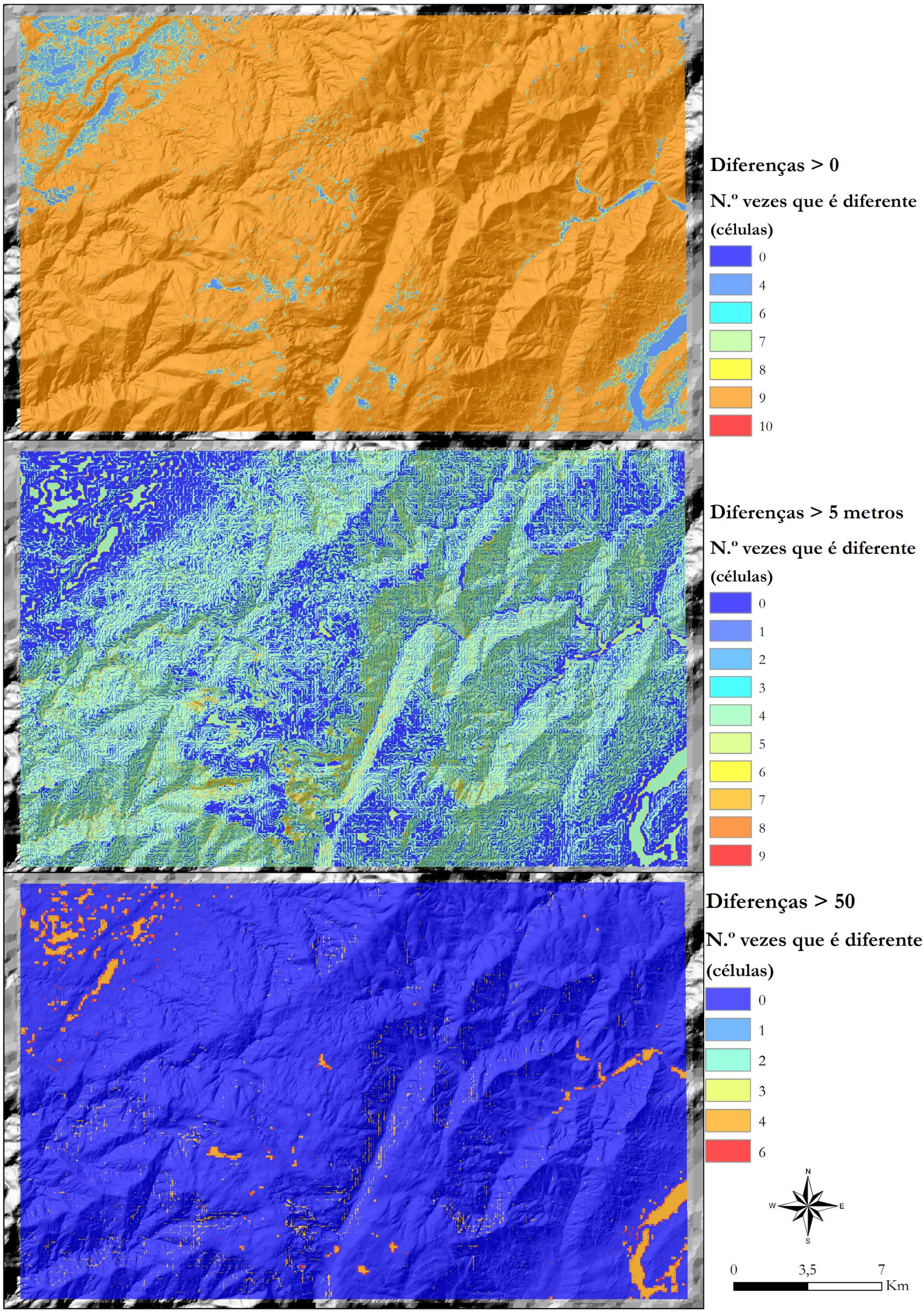


Figura 81 - Cartogramas que registam o número de vezes que uma célula foi considerada como diferente na subtracção entre os vários pares de imagens produzidos no decorrer deste exercício.

Passando à segunda fase deste exercício, que, recorde-se, tem como propósito de raiz, a equação de um cenário hipotético, no qual se procuram as melhores áreas para instalação de painéis fotovoltaicos, uma vez gerado o MDT (segundo este método de interpolação ou outro), estamos em condições de, a partir dele, derivar as exposições das vertentes.

Burrough & McDonnell (2000), com base noutros autores, referem e hierarquizam, em função da sua qualidade, vários métodos que têm como propósito a derivação de cartas de declive e de exposições de vertente. Aqueles autores apontam os métodos de Zevenbergen & Throne (1987) e de Horn (1981) como sendo os que produzem erros de exactidão residuais, sendo, por isso, utilizados em diferentes SP, entre os quais o ArcGIS da ESRI.

Através da leitura da parte do manual do ArcGIS relativa ao funcionamento da ferramenta *Aspect* do *Spatial Analyst Tools* > *Surface*, e do seu contraponto com a obra de Burrough & McDonnell (2000), percebemos que este *software* aplica o método de Horn (1981).

Assim sendo, utilizando um MDT construído pelo Programa 16[233], e não recorrendo à ferramenta IDW, gerámos as exposições em diferentes *software* que também aplicam o método de Horn (1981), recorrendo às seguintes ferramentas:

i. ArcGIS 10.2 – *Arctoolbox* > *Spatial Analyst Tools* > *Surface* > *Aspect;*
ii. gvSIG 2.2 – Sextante > *Geomorphometry and terrain analysis* > *Aspect;*
iii. GRASS GIS 7.0.0 – *r.slope.aspect* e posterior conversão para graus azimutais;
iv. SAGA GIS 2.1.2 – *Tool Libraries* > *Terrain Analysis* > *Morphometry* > *Slope, Aspect, Curvature;*
v. GDAL integrada no QGIS 2.10 – *Processing* > *GDAL/OGR* > *GDAL Analysis* > *Aspect.*

A Tabela XLVII documenta os resultados obtidos a partir do cálculo da correlação linear entre os vários resultados cartográficos[234]. Ao contrário do ensaio anterior, em que não havia valores de correlação de "1" (embora estivessem muito próximos da unidade), neste exercício destaca-se apenas um *output* que evidencia algumas diferenças relativamente aos restantes, falamos do gvSIG. À primeira vista, os resultados relativos ao GRASS pareciam também apontar algumas discrepâncias muito significativas, mas, viemos nós a perceber, que se explicam pelo simples facto de a ferramenta *r.slope.aspect* não apresentar a exposição das vertentes expressa em ângulos

[233] **Ver Anexo A, p. A13.**
[234] Sabemos que a exposição das vertentes é um atributo cíclico, em que os valores representam uma categoria e não um valor quantitativo que permita dizer, por si só, que um é mas importante que o outro. No entanto, salientamos mais uma vez que a aplicação da correlação linear tem como objectivo a averiguação de quão idênticas são as várias cartas de exposições, e nada mais.

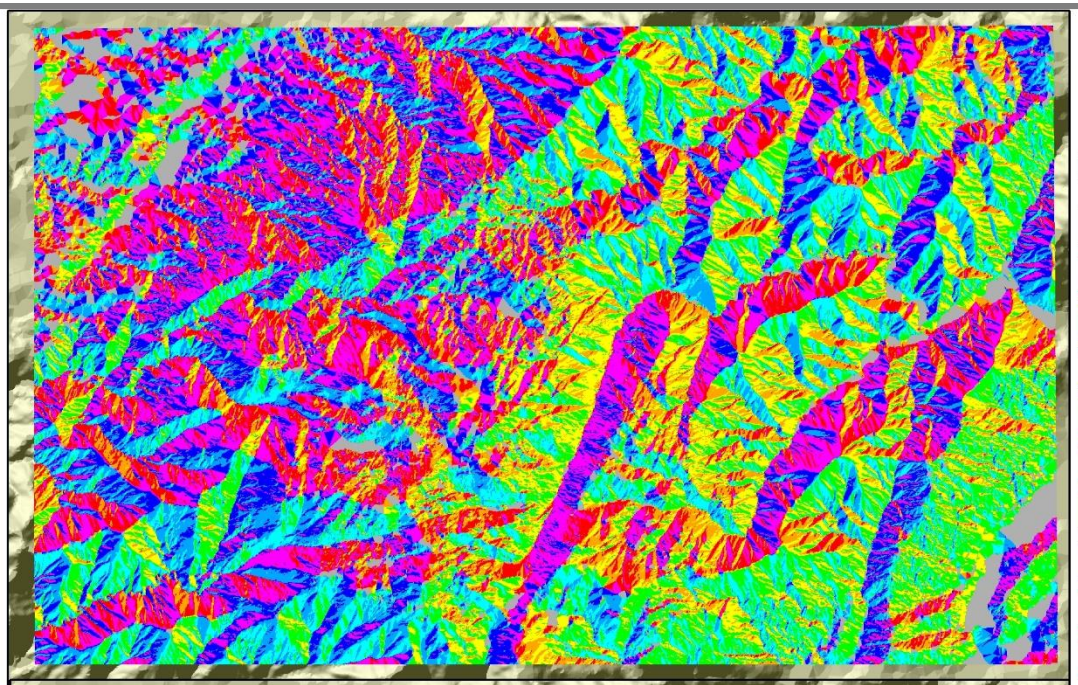
azimutais. Pelo contrário, o mapa gerado por esta ferramenta representa as categorias de exposição em número de graus de este, tal como é explicado na sua descrição (https://grass.osgeo.org/grass71/manuals/r.slope.aspect.html, acedido em 15 Fevereiro, 2016). Assim, para que a comparação entre o *output* gerado pelo GRASS e os *outputs* dos restantes fosse válida, em primeiro lugar, tiveram de se converter os valores de cada célula em graus azimutais, segundo a seguinte expressão: $grau_{azimutal} = (450 - célula_{xij})\ \%\ 360$. Esta conversão implica uma estratégia para preservar os valores das células que representam a categoria "plano" (ângulo 0 a partir de este), de modo a assegurar que estas áreas não venham a apresentar ângulos azimutais que as façam integrar a classe "este".

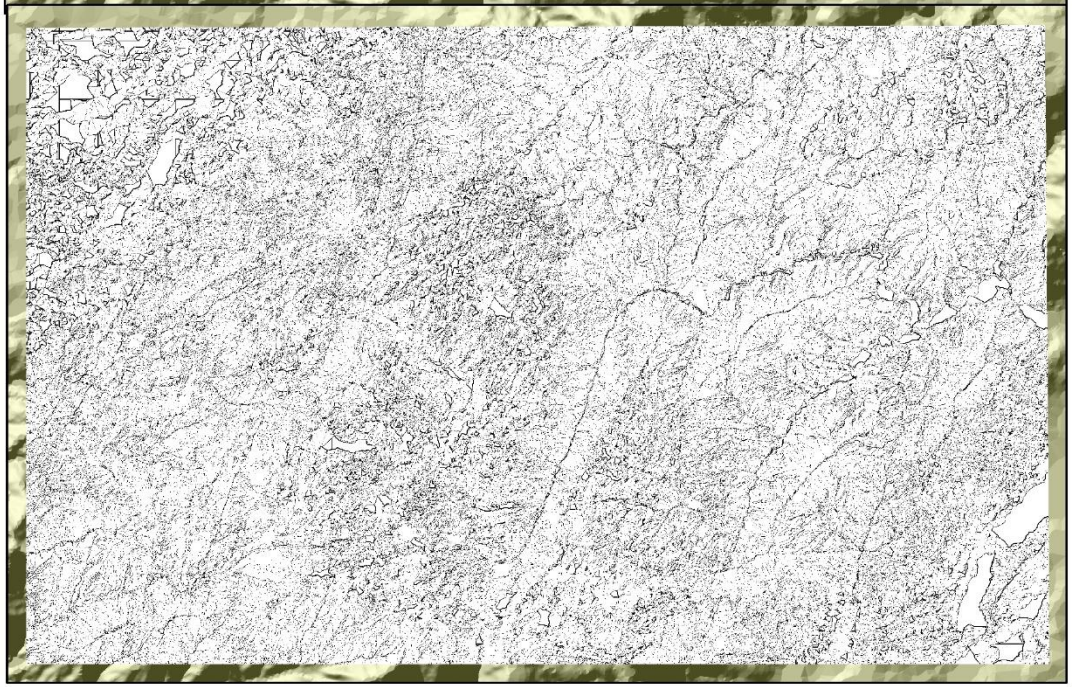
Tabela XLVII - Matriz de correlação entre os temas de exposição de vertentes gerados segundo o método de Horn (1981) em vários *software* SIG

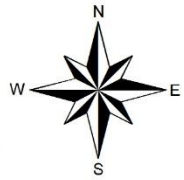
	ArcGIS 10.2	gvSIG 2.2	GRASS 7.0.0	SAGA 2.1.2
ArcGIS 10.2	**1**			
GvSIG 2.2	0,88	**1**		
GRASS 7.0.0	1	0,88	**1**	
SAGA 2.1.2	1	0,88	1	**1**
GDAL	1	0,88	1	1

A comparação das exposições geradas pelo gvSIG com as restantes imagens revelou uma distribuição ligeiramente diferente das dos restantes *software*, tal como se pode verificar nos cartogramas da Figura 82, que contrapõem, por exemplo, as exposições executadas em ArcGIS com as diferenças entre estas categorias e as produzidas pelo gvSIG. Por alguma razão, que não conseguimos identificar, existe um desfasamento que classifica de modo diferenciado 12,8% (792 856) dos pixéis da área de estudo. Estas distorções distribuem-se por toda a área de teste, não sendo possível identificar, sem uma análise pormenorizada que extravase a mera interpretação visual da Figura 82, um padrão que nos permita dizer em que tipos de áreas estas distorções ocorrem.

Facilmente se compreende que, se o operador não se preocupar com estas questões, necessariamente, há erros, mais ou menos graves que podem influenciar, de forma muito significativa o tipo de cartografia delas resultante, consoante o teor de exactidão que pretendamos Neste caso específico, por efeito de alguma irregularidade no código-fonte da ferramenta *Aspect* da *toolbox* Sextante*,* ou por qualquer outra particularidade deste *software* que não conseguimos descortinar, poderíamos acabar por instalar painéis fotovoltaicos em locais não tão favoráveis.







Exposições

- Plano (-1)
- Norte (0-22.5)
- Nordeste (22.5-67.5)
- Este (67.5-112.5)
- Sudeste (112.5-157.5)
- Sul (157.5-202.5)
- Sudoeste (202.5-247.5)
- Oeste (247.5-292.5)
- Noroeste (292.5-337.5)
- Norte (337.5-360)

Figura 82 - Demonstração das diferenças entre as cartas de exposições geradas por dois *software* distintos que usam o mesmo método para o efeito.

Diferenças apresentadas pelo gvSIG

- Igual
- Diferente

0 3,5 7 Km

No entanto, e pela perspectiva positiva da utilização de outro SL/CA destacam-se o GRASS GIS, o SAGA GIS e a biblioteca de geoalgoritmos GDAL/OGR, que apresentaram resultados quase indistintos dos da grande referência do mercado SIG. O SAGA GIS foi mesmo aquele que, nos dois exercícios, gerou resultados mais próximos aos do ArcGIS.

Ainda sobre os resultados deste exercício, não podemos deixar de recordar a Figura 71 (p. 244), à qual recorremos anteriormente para demonstrar que quando se realiza uma migração para SL/CA SIG, temos de estar atentos e ser receptivos a adaptações nas ferramentas análogas àquelas que costumamos utilizar, tomando conhecimento de novos parâmetros, características e modificações que possam existir, e evitando assumir que todas as ferramentas análogos trabalham os dados e expressam os resultados da maneira a que estamos habituados. Este exercício volta a ser uma importante chamada de atenção para isso, visto que a ferramenta *r.slope.aspect* do GRASS GIS não devolve um *output* em graus azimutais, à semelhança, por exemplo, do ArcGIS. Enfim, o operador tem de estar atento a estas mudanças, na medida em que o seu desconhecimento pode acarretar produções cartográficas com distorções mais ou menos significativas e importantes. Ao mesmo tempo, estas cautelas evitam que se diga que aquela ferramenta tem problemas, apesar de, na realidade, o problema ser do operador, que não conhece a ferramenta com que trabalha e comete erros atribuindo responsabilidades a terceiros.

Em jeito de conclusão, alguma literatura da especialidade refere-se à existência de inconformidades de resultados, designadamente, com expressão cartográfica, que podem decorrer da aplicação de geoprocessos, aparentemente iguais, ferramentas ditas análogas, mas que podem produzir resultados diferentes. Em alguns casos específicos, esta constatação tem algum fundamento, persistindo ainda alguns fantasmas relativos ao código que meros operadores de SIG, sem conhecimentos profundos nas mais variadas linguagens de programação, não conseguem compreender ou justificar, e, por isso, não conseguem corrigir potenciais erros, que podem ser cumulativos. Estes operadores estão, por isso, dependentes da sua capacidade de interpretação visual e estatística dos *outputs*.

Contudo, o exercício desenvolvido ao longo deste capítulo, pela sua simplicidade e desvinculação face a qualquer leitura mais parcial, não deve ser alvo de generalização, visto que apenas se consideraram o desempenho e os resultados visuais e estatísticos obtidos a partir de um conjunto restrito de algoritmos relacionados com a detecção remota e a modelação do relevo. Não é, por isso, aconselhável afirmar sem reservas que as diferenças aqui enunciadas se verificam com a execução de outras ferramentas, até porque, como vimos anteriormente, foram obtidos resultados idênticos nos extensos processos de modelação efectuados nos dois capítulos anteriores (e sempre

que tal não aconteceu, conseguimos identificar a razão para o sucedido). Não obstante, a presente exposição confirma e reforça os alertas, a que já nos referimos antes, em relação à necessidade incontornável de se promover um escrutínio que ateste a integridade lógica e matemática dos geoprocessos que, de forma recorrente, usamos nas nossas actividades e projectos em SIG, mas, alerta também, para a necessidade de promover uma postura crítica na análise dos resultados debitados pelos diferentes *software* utilizados.

3ª Parte –

Urgência e emergência da SIGósAPp – conceito e viabilidade de um programa para equação (automática) dos custos para o modelo proprietário e para o modelo livre e/ou de código aberto

CAPÍTULO I – ADOPÇÃO DE *SOFTWARE* LIVRE E DE CÓDIGO ABERTO – UNIVERSO BIBLIOGRÁFICO, QUE POSSIBILIDADES DE INVESTIGAÇÃO?

A revisão de literatura é sempre um processo que visa a orientação epistemológica das hipóteses de investigação, provando a pertinência do objectivo do investigador. Percebendo o que já foi feito e o que falta fazer no âmbito da formulação de um modelo relacionado com a adopção de SL/CA em contexto organizacional, poderemos demonstrar a inexistência e a viabilidade que teria uma aplicação como a SIGósAPp.

Tendo em vista a concretização dos desideratos gerais desta dissertação relacionados com a concepção da SIGósAPp, delineou-se um périplo (Figura 83) que nos levaria, da teoria à prática, desde os fundamentos teóricos que sustentam o problema da adopção da inovação e gestão da mudança na organização até à implementação de plataformas que visam a comparação directa entre *software* e, no seu seguimento, do MDS4OSS (com enfoque no *software* SIG), a desenvolver em trabalho futuro.

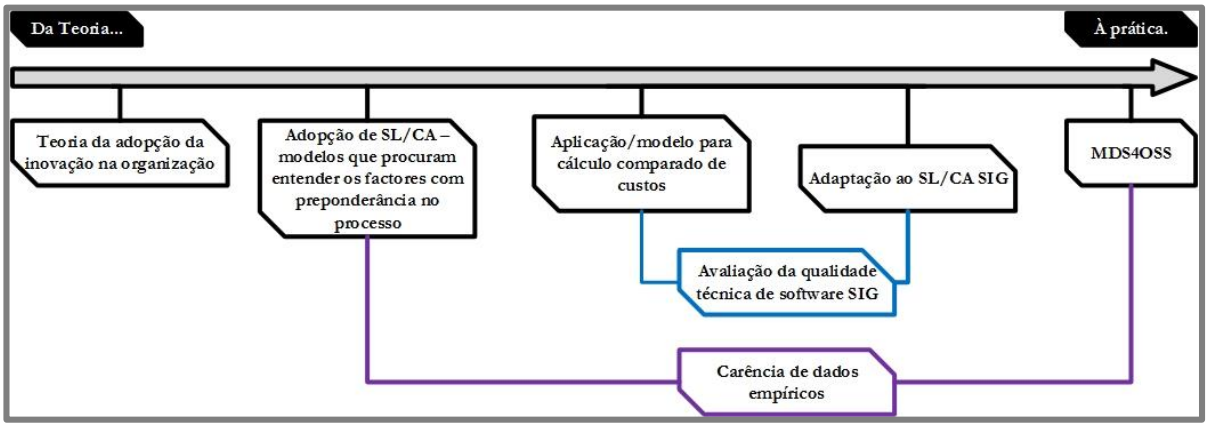


Figura 83 - Linha de pesquisa sobre as possibilidades de investigação no âmbito da concepção e concretização da SIGósAPp.

Como corolário deste contexto e das nossas pretensões iniciais, no presente capítulo vamos colocar o foco da nossa abordagem nos seguintes tópicos e respectivas metas:

1. **Adopção de inovações tecnológicas em contexto organizacional: bases terminológicas e conceptuais para conceber um modelo de adopção**
    i. Elencar um conjunto de autores e obras que abordam e concebem a Teoria da Adopção da Inovação, suporte bibliográfico da SIGósAPp.
2. **A adopção de SL/CA em contexto organizacional – características da literatura de referência**
    i. Inventariar referências e conhecimentos sobre matérias relacionadas com a implementação/adopção de SL/CA, nomeadamente, na administração pública, de modo a conhecer o que já se fez no âmbito da produção de um guia/manual de apoio ao balanço de custos (entre soluções informáticas) para efeitos de se considerar a opção "migração".
3. **Ancestralidade e linhagem da SIGósAPp: das metodologias analógicas para equacionar soluções à automatização da decisão**
    i. Catalogar aplicações/plataformas existentes para o balanço de custos entre soluções proprietárias e livres e de código aberto, percebendo quais são os principais indicadores envolvidos em modelos práticos de cálculo do TCO e ROI.
4. **Síntese: primórdios da formulação de um guia/manual de apoio à decisão e migração com enfoque no *software* SIG *desktop***
    i. Sintetizar tudo o que se tiver dito até aqui.

I. A adopção de inovações tecnológicas em contexto organizacional: bases terminológicas e conceptuais para conceber um modelo de adopção

> "*Uma das razões pelas quais estamos interessados na difusão das inovações prende-se com o facto de ser extremamente difícil adoptar uma nova ideia, mesmo que apresente vantagens óbvias. Existe uma grande brecha (em muitos campos) entre o que se sabe e o que está actualmente a ser posto em utilização. Muitas inovações requerem um longo período, de alguns anos, desde o momento em que ficam disponíveis até ao momento em que são largamente adoptadas. Por isso, um problema comum para muitos indivíduos e organizações tem a ver com a velocidade a que se dá a difusão de uma inovação.*"
>
> Everett Rogers (1983:1)[235]

A implementação de uma nova solução informática (inovação) numa estrutura institucional não é um processo trivial nem pacífico e, merece, por isso, planeamento, estratégia e reflexão cuidada, tal é a complexidade e o risco que envolve (Dodgson *et al.*, 2008). Não é por acaso que existe uma linha de investigação que se preocupa exclusivamente com esta problemática (Rogers, 1983; Depietro *et al.*, 1990 *apud* Ven & Verelst, 2011; Tortnakzy *et al.*, 1990; Fichman, 1992; Poole *et al.*, 2000; Frambach & Schillewaert, 2002; Dodgson *et al.*, 2008; Kulviwat *et al.*, 2009; Talukder, 2012).

Numa perspectiva holística, enveredar pelo estudo da gestão da mudança implica cuidados metodológicos precisos e algo distintos de outras abordagens unidisciplinares (Rodrigues, 1998); trata-se de uma longa mudança que deverá passar por uma sequência de etapas-chave, assim como as apresentadas por Galpin (2000). A gestão da inovação tecnológica, enquanto um tipo de mudança em contexto organizacional, ultrapassa as balizas da mera aposta tecnológica, pois, em última análise, as tarefas são executadas por pessoas (Pfeffer, 1995 *apud* Rodrigues, 1998), logo, na

[235] Tradução livre a partir do original, "*One reason why there is so much interest in the diffusion of innovations is because getting a new idea adopted, even when it has obvious advantages, is often very difficult. There is a wide gap in many fields, beteween what is know and what is actually put into use. Many innovations require a lengthy period, often of some years, from the time when they become available to the time when they are widely adopted. Therefore, a common problem for many individuals and organizations is how the speed up the rate of diffusion of an innovation". In* ROGERS, Everett (1983) – **Diffusion of Innovation**. The Free Press, 3ª Edição, Nova Iorque, 453 p.

implementação de SL/CA, a aposta na comunicação, numa boa estratégia de gestão da cultura organizacional e aplicação de bons cânones de liderança são fulcrais para a superação da resistência e sustento do *empowerment* da equipa (Galpin, 2000).

Porém, antes da implementação efectiva da tecnologia, a teoria da difusão e adopção da inovação (Rogers, 1983) ou teoria da Tecnologia-Organização-Ambiente (Depietro *et al.*, 1990 *apud* Ven & Verelst, 2011) aponta para diferentes etapas que visam a constituição do *innovation-decision process*, processo através do qual uma organização toma conhecimento sobre a inovação (pela difusão[236]), tomando uma atitude relativamente a ela, no sentido de a adoptar ou rejeitar, consoante determinados factores, mormente os relacionados com o custo, aspectos técnicos, sociais, organizacionais e com experiências tidas por outros.

Em suma, estas breves considerações demonstram que a adopção da inovação é um sistema compreendido entre a percepção da existência da novidade tecnológica e a gestão humana da mudança. Estes pressupostos teóricos têm um papel fundamental na orientação da elaboração da SIGósAPp, na medida em que nos auxiliam a formular e aplicar uma análise de cálculo efectivo do custo de cada solução baseada em factores organizados em quatro grandes grupos, indo ao encontro das propostas de autores como Frambach & Schillewaert (2002); Glynn *et al.* (2005); Fitzgerald (2011); Qu *et al.* (2011); Ven & Verelst (2011); Bouras *et al.* (2013):

i. Factores tecnológicos;

ii. Organizacionais (internos ou sociais);

iii. Económicos;

iv. E externos.

A qualidade técnica e os baixos custos não são suficientes, a aceitação individual e influência social são variáveis chave para o processo (Frambach & Schillewaert, 2002; Kulviwat, 2009), assim se justifica porque é que a estratégia de gestão do capital humano na mudança, *a montante* e *a jusante* da decisão, é tão importante e deverá constar obrigatoriamente no MDS4OSS (Rodrigues, 1998; Galpin, 2000; Dodgson *et al.*, 2008).

[236] Entende-se difusão como o processo pelo qual uma inovação tecnológica, enquanto ideia, prática ou objecto é percebida como nova por um indivíduo ou outra unidade de adopção potencial como uma organização pública ou privada, e que é comunicada (transmissão especial em que as mensagens são novas ideias, partilhadas entre todos para um entendimento mútuo) por certos canais do sistema social ao longo do tempo (Rogers, 1983).

2. A adopção de SL/CA em contexto organizacional – caracterização da literatura de referência

> *"Existem poucas publicações que apresentem dados empíricos sobre a adopção de software de código aberto, como se não bastasse, a qualidade dos trabalhos publicados não é boa o suficiente."*
>
> Øyvind Hauge; Claudia Ayala & Reidar Conradi (2010:1146)[237]

Scacchi *et al.* (2006), Krogh & Hippel (2006), Østerlie & Jaccheri (2007), Stol & Babar (2009) e Hauge *et al.* (2010) denunciam o baixo número de referências que abordam o tema da adopção de SL/CA em contexto organizacional. Entre 1998 e 2008, de um universo de publicações sobre SL/CA (Figura 84), apenas uma percentagem pouco significativa (por ano) trata o problema da sua adopção em instituições e, uma parte considerável delas, não apresenta qualquer tipo de informação empírica digna de destaque (Hauge *et al.* 2010)[238].

Na realidade, este tipo de estudo tem merecido pouca atenção por parte dos investigadores, isto porque o SL/CA exige uma metodologia de validação do esquema de adopção única, completamente distinta do modelo proprietário por envolver um conjunto de factores específicos e por merecer uma problematização particular, caso a caso (Frambach & Schillewaert, 2002; Glynn *et al.*, 2005; Østerlie & Jaccheri, 2007; Stol & Babar, 2009; Wise, 2012; Bouras *et al.*, 2013; Marson & Paré, 2013). Acresce ainda que, as publicações existentes são de qualidade duvidosa, nomeadamente, as que dizem contribuir com dados empíricos sobre esta matéria (Hauge *et al.*, 2010)[239].

---

[237] No original, "*There are relatively few empirical publications on OSS in organizations, and the quality of published work is not good enough.*" *In* HAUGE, Øyvind; AYALA, Claudia & CONRADI, Reidar (2010) – "Adoption of open source *software* in *software*-intensive organizations – A systematic literature review". **Information and *Software* Technology,** N.º 52, 1133-1154.

[238] Hauge *et al.* (2010) analisaram, entre 1998 e 2008, 24 289 publicações distribuídas por 24 periódicos e 7 conferências. Destas, apenas 112 *papers* apresentam evidências empíricas sobre como as organizações implementam SL/CA.

[239] Segundo Hauge *et al.* (2010), os 112 estudos que integram dados empíricos revelam várias limitações: falta de um enfoque particular (tópicos generalistas); escassos exemplos de relatos empíricos sobre adopção; qualidade limitada; pouca relevância (abordam o fenómeno livre, mas não tratam problemas específicos que aqueles interessados têm para resolver, pois não fornecem recomendações práticas em número suficiente; imaturidade dos dados; não reflectem o contexto onde o SL/CA é implementado; e não estimulam campos de pesquisa relacionados com o fenómeno.

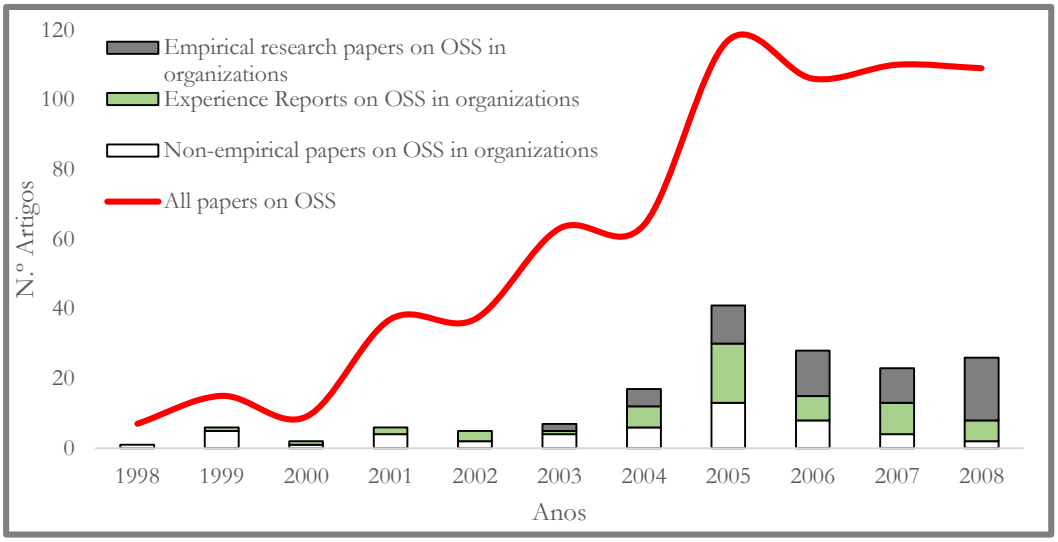


Figura 84 - Número de publicações sobre SL/CA e SL/CA em organizações. Fonte: Hauge *et al.* (2010), p. 1140.

A revisão de literatura que promovemos acaba por confirmar isso mesmo. A leitura de vários documentos permitiu-nos a definição de três categorias segundo o nível de experiência empírica que transmitem:

i. Trabalhos que problematizam os factores potenciadores ou inibidores de adopção sem a utilização de processos estatísticos que expliquem a sua relevância no processo de implementação;

ii. Concepções que associam a adopção de SL/CA à sua relação puramente estatística com factores que potenciam ou inibem a adopção;

iii. Trabalhos/plataformas que fornecem experiências (reais) relacionadas com a adopção de SL/CA e que propõem metodologias de base empírica para ponderação de factores de adopção definidos como determinantes.

2.1. Trabalhos que problematizam os factores potenciadores ou inibidores de adopção sem a utilização de processos estatísticos que explicam a sua relevância no processo de implementação

Kovács *et al.* (2004) expõem as suas dúvidas relativamente à adopção de SL/CA e de normas abertas na administração pública europeia, enunciando alguns exemplos de implementações mais ou menos bem-sucedidas (Itália, Alemanha, Bélgica ou Irlanda), sobre as quais discutem alguns aspectos positivos e negativos. Hwang (2005) cita alguns factores que influenciam o movimento

em direcção ao SL/CA, tidos como preponderantes e fundamentais. Encontra-se o mesmo em Perry & Margoni (2010) e em Oram (2011), nos quais se alteram apenas os exemplos dados e o enfoque.

Outros, como Kwan & West (2005), discutem a conceptualização de um modelo teórico de adopção de SL/CA em contexto organizacional, enunciando as várias dimensões que o integram. Embora se faça referência a um exemplo real de implementação, a exposição é meramente teórica.

Grosso modo, tratam-se de referências meramente especulativas que problematizam as vantagens do SL/CA na administração pública. Embora os discursos sejam variados, os argumentos são bastante semelhantes. Por conseguinte, o índice de redundância é elevado.

2.2. Concepções que associam a adopção de SL/CA à sua relação puramente estatística com factores que potenciam ou inibem a adopção

Trata-se de correntes de pensamento que abandonam a simples problematização de factores potenciadores e inibidores da migração. Estes trabalhos (bem como os seus autores) partem para uma abordagem estatística que, apoiada na teoria da difusão e adopção da inovação, procura perceber o maior ou menor peso de uma determinada variável no processo de adopção/migração, mediante a consideração de uma amostra real de casos. Infelizmente, esta abordagem não coloca "a nu" a relação entre os resultados estatísticos e as razões (dificuldades sentidas) por detrás do sucesso ou insucesso de um processo de migração.

Na escassa literatura da especialidade encontramos, todavia, alguns exemplos de trabalhos onde se apresenta a execução de um conjunto de processos estatísticos que tentam compreender a relação/correlação directa entre a adopção/migração e um conjunto de factores que, segundo outros autores (por eles consultados e referidos), têm forte implicação no processo (modelo de adopção), confirmando ou não um conjunto de hipóteses estabelecidas *a priori* (Glynn *et al.*, 2005; Qu *et al.*, 2010; Ven & Verelst, 2011; Spinellis & Giannikas, 2012; Engelhardt & Freytag, 2013, entre outros).

Por exemplo, no artigo de Glynn *et al.* (2005), o quadro de suporte à investigação da adopção da inovação, que esquematizamos na Figura 85, foi validado a partir da experiência real retirada de um

objecto de estudo (hospital) que enveredou pela implementação de SL/CA em significante parte da sua infra-estrutura de TI, diagnosticando-se a validade das hipóteses previamente definidas[240].

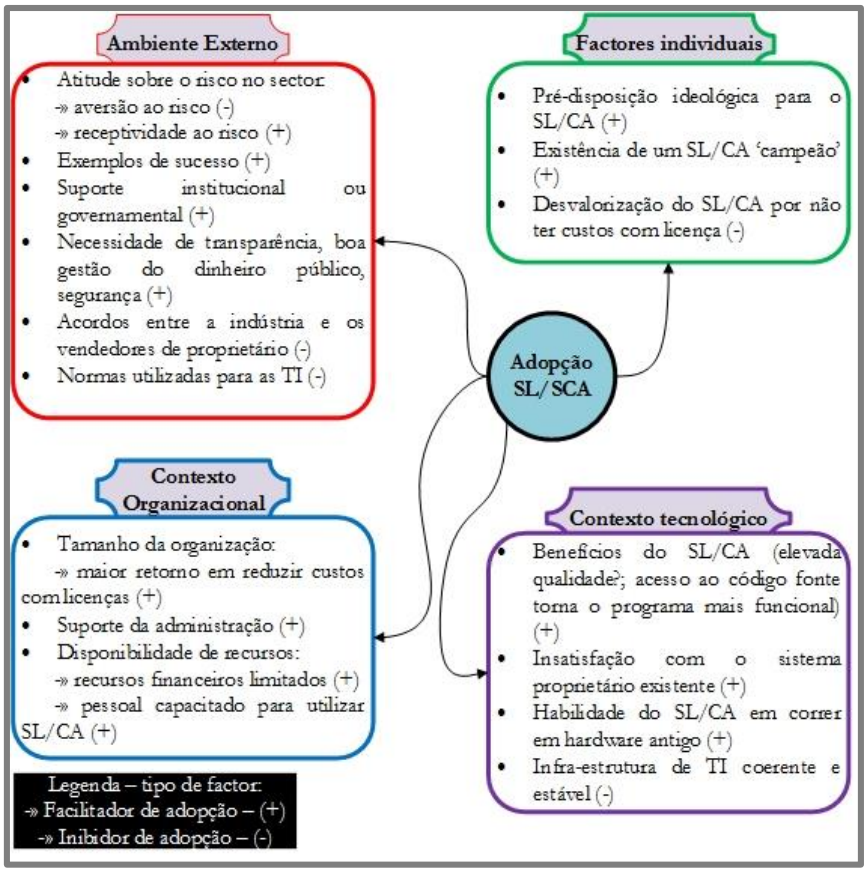


Figura 85 - Quadro para a investigação da adopção de SL/CA. Fonte: Glynn *et al.* (2005), p. 226.

No trabalho de Qu *et al.* (2010), os autores definiram duas escalas de análise – nível da empresa e nível do país – e três conjuntos de variáveis que procuravam explicar a variável dependente – um grupo que explica a adopção a nível empresarial, outro adaptado à escala do país, e um último onde se incluem variáveis de controlo do modelo estatístico (Figura 86) [241].

[240] A validação realizou-se mediante a análise das respostas a um questionário, colocado a um grupo de indivíduos ligados ao hospital.

[241] Os nomes das variáveis apresentadas na Figura 86 podem parecer algo ambíguos, pelo que se justifica o seguinte esclarecimento. Segundo Qu *et al.* (2011):

i. Na teoria da difusão da inovação, a difusão é transportada através de canais de comunicação (Rogers, 1983), logo, esta será influenciada pelas características das redes de contactos que se estabelecem entre os diferentes organismos (públicos ou não). Assim, a adopção de SL/CA numa empresa depende do facto de estabelecer relações com uma rede composta por instituições que trabalham com *software* proprietário ou por uma rede constituída por órgãos que já usam SL/CA.
ii. O nível de incerteza refere-se ao modo como as pessoas se sentem confortáveis ou não com aquilo que é novo, desconhecido, surpreendente ou incomum.
iii. A distância ao poder refere-se à forma como os membros menos poderosos da sociedade esperam e aceitam uma distribuição de poder desigual.

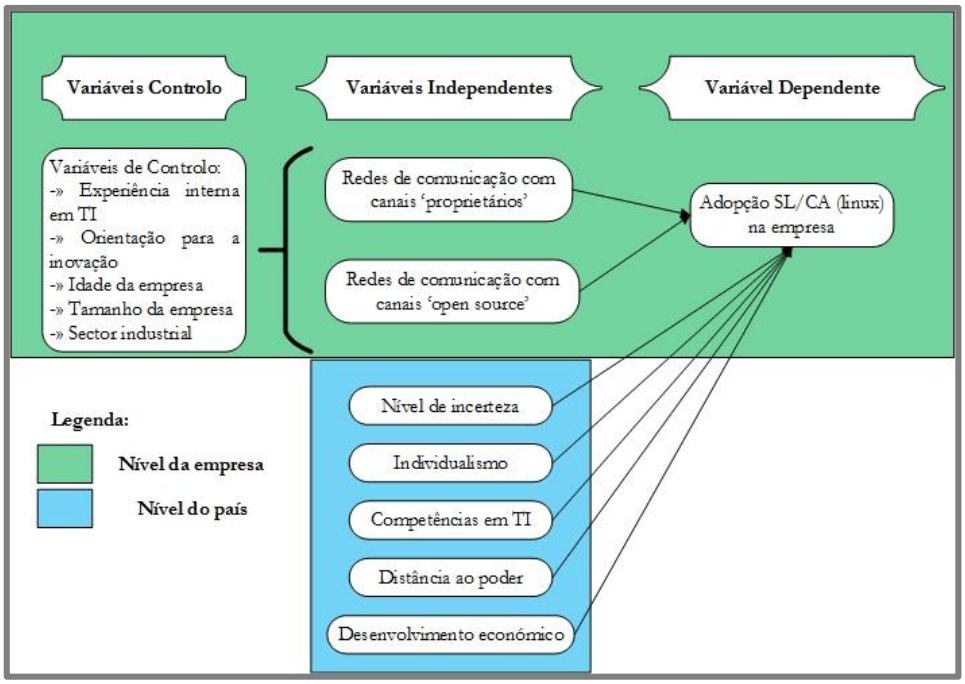


Figura 86 - Modelo multi-nível de adopção de SL/CA. Fonte: Qu *et al.* (2011), p. 1001.

Para além destes, destacamos mas referimos de modo telegráfico os trabalho de:

i. Engelhardt & Freytag (2013), que se debruça em específico sobre o impacto de factores culturais e institucionais na adopção de SL/CA.

ii. Subramanyam & Xia (2008), trabalho no qual se utilizaram vários métodos estatísticos para identificar, em três regiões do globo (América do Norte, China e Índia), quais as motivações genéricas e preferências que têm maior relevância quando se participa em projectos de SL/CA.

iii. Gallego *et al.* (2015), um trabalho que procurava medir a preponderância do treino/prática/experiência técnica com SL/CA enquanto factor de adopção, adiantando-se a necessidade de implementação de actividades educacionais que promovam a formação de pessoal especializado neste tipo de solução, grupo que constituirá suporte e garantias de sucesso.

iv. O individualismo corresponde ao grau de integração em grupos.
v. As competências em TI dizem respeito ao grau de experiência e capacidades do pessoal ao serviço em matéria de *software* e ao nível de suporte tecnológico que uma empresa pode esperar nesse país.
vi. Sobre o desenvolvimento económico, os autores colocam a hipótese de que um país mais desenvolvido economicamente terá menor necessidade de adoptar SL/CA, soluções procuradas maioritariamente por aqueles com menos recursos.

iv. Marsan *et al.* (2012a), que apresentou um estudo sobre um conjunto de instituições que já iniciaram o seu processo de transição e as principais razões que as levaram a essa tomada de decisão, percebendo-se que, em 2009, a institucionalização de SL/CA já tinha sido alcançada com sucesso para aplicações do tipo *back-end* (*software* que suporta o trabalho do computador, sistema e especialistas), mas que em aplicações *front-end* (*software* que é usado para dar suporte aos utilizadores do sistema), o processo ainda estaria apenas em fase embrionária. Os autores concluíram também que este processo já tinha começado em 2005, registando uma aceleração considerável entre 2006 e 2009.

v. Por fim, Marsan *et al.* (2012b) abordam estatisticamente a percepção que os especialistas em TI têm do discurso público sobre SL/CA e a relação que isso tem com a abertura das suas organizações a este tipo de *software*.

Em Portugal, sobre adopção de SL/CA, apenas servem de exemplo os estudos de Fernandes (2010) e Godinho (2012), encaixados nesta categoria[242]. Em ambos, colocam-se algumas hipóteses que se confirmam ou não através da interpretação estatística das respostas de um inquérito por questionário, à semelhança de Glynn *et al.* (2005).

Nestes exemplos discutem-se e propõem-se factores, utilizam-se outras escalas, e os mais variados métodos estatísticos, mas, no geral, repetem-se os objectivos deste tipo de estudo. Ao contrário das referências da primeira categoria, estas representam um avanço significativo na transposição deste problema, da teoria, para a prática, visto que apuram algumas das principais variáveis (e as que devem merecer maior ponderação) que devem integrar o cálculo do TCO. Porém, mesmo que na totalidade dos casos, se apoiem em exemplos específicos e reais, à semelhança dos outros tipos de trabalho, não facultam dados empíricos substanciais sobre o que possa ter decorrido melhor ou pior.

2.3. Trabalhos/projectos que fornecem experiências (reais) relacionadas com a adopção de SL/CA e que propõem metodologias de base empírica para ponderação de factores de adopção definidos como determinantes

Os dados empíricos são substanciais para a reunião de um grupo de variáveis preponderantes no processo de migração, como referem Marsan & Paré (2013, p. 737) "…não dispomos de exemplos

[242] O primeiro tinha como objecto de estudo a administração pública local portuguesa, o segundo debruçou-se sobre a administração central portuguesa. Estas correspondem ao transporte da *main stream* para Portugal, isto em termos de forma de investigar a adopção de SL/CA.

que nos sirvam de modelo a seguir de modo a que nos seja possível estar seguros das nossas opções". Só mediante a análise da experiência daqueles que viveram e geriram o processo de migração (lições, aspectos positivos e erros a evitar) é que se podem constituir indicadores que deverão ser integrados no balanço TCO/ROI, norteando a formulação do MDS4OSS.

Em Portugal, os dados disponíveis não vão muito além daquilo que já foi descrito, isto é, não há quaisquer informações (experiências vividas) relevantes e fundamentadas, de forma clara e coerente, sobretudo do ponto de vista procedimentar e da análise ponderada de factores, sobre a adopção de SL/CA, no geral, e de SL/CA SIG, em particular.

Na literatura internacional, são também escassos os exemplos. Ainda assim, destacam-se os trabalhos de Dedrick & West (2004), Fitzgerald *et al.* (2011) e Marsan & Paré (2013), que adoptaram uma abordagem que coloca em evidência a experiência e a opinião pessoal dos entrevistados (um exercício que também implementámos, como antes se referiu, mas cujo limitado sucesso não nos autorizou a retirar conclusões substanciais sobre o assunto). Os primeiros autores acima referidos, preocupam-se com a adopção do Linux num grupo de organizações, problematizando as suas características e as opiniões dos entrevistados, confrontando-as com os factores normalmente associados a um modelo de adopção (factores tecnológicos, organizacionais e externos), de modo a discutirem os pontos fortes e fracos da tecnologia naquelas entidades.

Inspirados pela Teoria da Adopção da Inovação de Rogers (1983), os segundos privilegiaram uma abordagem de descrição exaustiva de cinco casos concretos e específicos de migração para SL/CA e adopção de *open standards* (relacionados com ferramentas de produtividade e escritório) distribuídos por todo o mundo. Da leitura destas páginas resultam lições sobre o modo como os funcionários reagiram ao processo de migração, bem como outras ilações importantes sobre estratégias gerais a adoptar quando se implementa SL/CA.

Por fim, Marsan & Paré (2013) tomam como exemplo a adopção de SL/CA nos serviços de saúde do Quebec (Canadá). Das entrevistas realizadas a *IT managers* de organizações de saúde ou a outros que fornecem *software* e serviços contratualizados por essas mesmas organizações, resultaram conclusões pertinentes, fruto da experiência dos entrevistados. Neste estudo, os autores averiguaram quais os factores com maior capacidade para potenciar ou inibir a adopção de SL/CA. Mas, ao contrário dos estudos elencados na secção imediatamente anterior, neste trabalho não

foram usadas quaisquer medidas estatísticas, pelo contrário, recorreu-se a análises empíricas que originam uma reflexão em torno das "boas" ou "más" práticas e experimentos[243].

Por sua vez, com o intuito de minimizar as dificuldades sentidas na migração, outros autores propõem várias metodologias que visam uma boa gestão do processo de transição para SL/CA.

Referimos a título de exemplo, o trabalho de Bouras *et al.* (2013), no âmbito do projecto OSEPA (*Open Source Software usage by European Public Administration*), que começam por definir um conjunto de factores que afectam o uso e a adopção de SL/CA (Tabela XLVIII), a partir dos quais formularam *guidelines* que, supostamente, deverão orientar os administradores públicos no processo de decisão que conduz à tomada de decisão sobre a opção de escolha entre SP e SL/CA (Figura 87).

Tabela XLVIII - Factores que influenciam o uso e a adopção de SL/CA. Adaptado de Bouras *et al.* (2013), p. 103

Factores tecnológicos:	Funcionalidade; suporte; manutenção; gestão; longevidade; confiança; disponibilidade; segurança; performance; usabilidade (interface entre o homem e a máquina – *user friendly*); interoperabilidade; flexibilidade e grau de customização; período de teste.
Factores organizacionais:	Habilidade de encontrar pessoal adequado e competente; falta de consciência; formação; resistência à mudança; forte liderança; suporte por parte da gestão; disponibilidade de conhecimentos dentro da organização; experiência em contexto real (outros exemplos); interoperabilidade entre aplicações.
Factores económicos (custo):	Custos do trabalho (eficiência); custos com licenciamento de *software*; controlo e aumento do acesso a propriedade intelectual; promoção do *software* utilizado no sector público.
Factores sociais:	Partilha de conhecimento; satisfação em se alcançar algo com valor; reputação profissional e reconhecimento entre os pares; aprender e melhorar competências pessoais; legalidade; sentimento de pertença à comunidade; melhoria dos produtos; liberdade de desenvolvimento.

Na Figura 87 apresenta-se um conjunto de etapas que respondem à obrigatoriedade de organização do raciocínio quando o objectivo é migrar ou comparar soluções, alertando-nos para a importância da selecção das soluções mais promissoras entre um universo alargado de alternativas. Trata-se de um (indispensável) primeiro passo, *a montante* da definição de um modelo funcional e prático para apoio à decisão sobre a escolha entre várias possibilidades.

[243] Destacam-se comentários sobre o custo do *software*, incompatibilidades de formatos, características da organização e recursos humanos ao sue serviço, falta de suporte, etc.

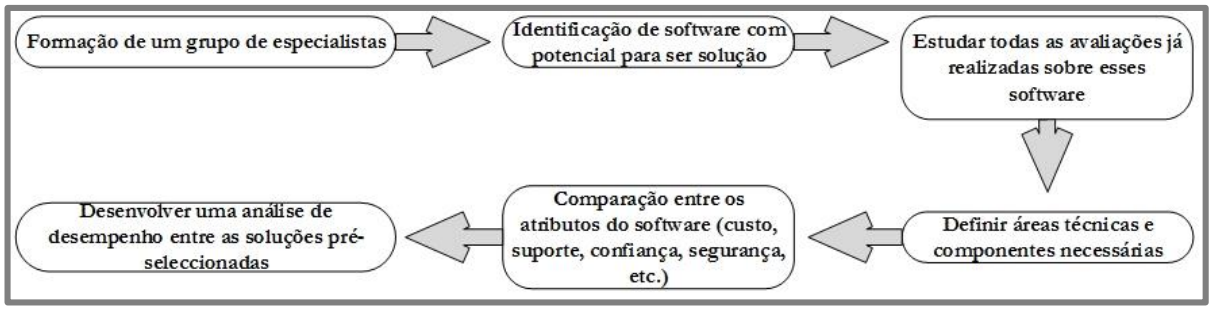


Figura 87 - *Guidelines* para selecção entre soluções livres ou de código aberto e proprietárias. Fonte: Bouras *et al.* (2013), p. 108.

Entre estas fases sequenciadas, a comparação e avaliação dos atributos do *software* é preponderante. Entre esses atributos, a componente técnica sobressai, como se tem vindo a salientar, os programas devem ser posto à prova. Recordamos o estudo de Sarrab & Rehman (2014) e a Figura 35, um verdadeiro exemplo de partilha de experiência empírica sobre a qualidade de um conjunto de aplicações para gestão de redes informáticas.

Dando seguimento a este périplo literário, há espaço agora para nos referirmos de modo sintético a alguns projectos como o OSEPA, que solicita particular atenção. Entre um universo de programas Europeus que promovem a implementação de SL/CA, como o PROSE[244] (*Promoting Open Source in European Projects*) ou o ISA[245] (*Interoperability Solutions for European Public Administrations*), o OSEPA foi um programa levado a cabo no âmbito da *EU National funded Interregional Cooperation Programme* (INTERREG IVC) entre 2010 e Dezembro de 2012, com o objectivo de melhorar o conhecimento e competências dos administradores públicos (locais e regionais) sobre soluções livres ou de código aberto, no que respeita à análise, troca e transferência de boas práticas, de modo a constituir um conjunto de estudos que definam a estratégia de migração mais eficaz (http://osepa.eu/index.php?em_cat=1, acedido 9 Junho, 2015).

Dado o elevado número de relatórios disponíveis no site oficial do projecto (http://osepa.eu/pdeliverables/index.php, acedido 9 Junho, 2015), não estamos em condições de os analisar de modo exaustivo, pelo que se seleccionaram alguns tópicos que nos parecem ter mais relevo e que poderão sustentar a elaboração da SIGósAPp ou de um MDS4OSS:

[244] O PROSE, ao contrário do OSEPA, executa medidas práticas de promoção da adopção através da criação e gestão de uma plataforma para gestão de projectos relacionados com SL/CA, e do desenvolvimento de um programa de treinos sobre os aspectos legais e de negócio relativos à adopção de SL/CA (http://www.ict-prose.eu/about/, acedido 9 Junho, 2015).

[245] Por seu turno, o ISA providencia suporte ao desenvolvimento de ferramentas e serviços na área do *e-Government*. Neste momento, constam já com mais de 40 ferramentas no seu palmarés, *software* livre, colocado à disposição de todos os interessados (http://ec.europa.eu/isa/ready-to-use-solutions/index_en.htm, acedido 9 Junho, 2015).

i. À semelhança de Bouras *et al.* (2013) (cfr. Figura 87), damos conta que existem vários documentos que tratam o problema da definição de um *adoption roadmap* (OSEPA, 2012a), a partir do qual se determinam conjuntos de etapas para avaliação (*benchmarks*) das soluções e sua adopção (OSEPA, 2012c; OSEPA, 2012d; OSEPA, 2012h; OSEPA, 2012i).

ii. Disponibilização de *checklists* para facilitar a tomada de decisão (OSEPA, 2012a).

iii. Inquirições que têm em vista percepcionar quais são as atitudes e barreiras que se colocam ao SL/CA e à sua implementação (OSEPA, 2012e).

iv. Exibição de estratégias e critérios para avaliação de boas práticas – de modo a perceber se um determinado caso de implementação é um caso de sucesso -, veja-se a *OSEPA Good Practice Evaluation Chart* (OSEPA, 2012a; OSEPA, 2012c) – metodologias fundamentais para a recolha de verdadeiros dados empíricos.

v. Recolha e formulação de considerações sobre a experiência individual de quem conhece as implicações da adopção de SL/CA – vantagens e desvantagens (OSEPA, 2012b).

vi. Caracterização das iniciativas legislativas em diversos países da União Europeia (OSEPA, 2012f).

vii. Recomendações sobre as políticas a seguir nestas matérias (OSEPA, 2012g; Bouras *et al.*, 2014).

viii. Definição de um conjunto de factores/variáveis que devem constar de um modelo de comparação entre soluções proprietárias e livres ou de código aberto (OSEPA, 2012h).

Chegamos, pois, à conclusão de que já existem manuais/guias de apoio à decisão e à implementação de SL/CA na administração pública, em países europeus. No entanto, ainda que o OSEPA nos forneça informações cruciais sobre como constituir um MDS4OSS, da análise dos documentos citados resulta a ideia de que estes *roadmaps* são ainda generalistas, na medida em que não abordam soluções concretas em contextos específicos.

3. Ancestralidade e linhagem da SIGósAPp: das metodologias analógicas para equacionar soluções à automatização da decisão

> "*Este projecto… permite que os utilizadores se registem e descubram projectos de SL/CA e explorem indicadores sobre a qualidade de uma maneira intuitiva, tendo em vista o auxílio da tomada de decisão. Adicionalmente, os utilizadores podem receber notificações quando os projectos, pelos quais se interessam, se posicionam abaixo (ou acima) de limiares seleccionados, relativos às métricas que mais interessam aos utilizadores.* "
>
> OSSMETER (2015)[246]

Anteriormente, referiram-se abordagens metodológicas que tinham como propósito perceber quais são as etapas e os factores mais preponderantes na avaliação comparada entre SP e SL/CA. A partir do que foi dito depreende-se que, mesmo havendo um *software* que compare duas ou mais soluções, não é permitido ignorar a necessidade de um planeamento, ou seja, será sempre necessário um esforço de estudo da viabilidade da mudança e um modo de gestão da mesma. Não obstante, uma ferramenta para avaliação comparada de soluções pode ser uma mais-valia para o decisor. Neste sentido, passamos em revista as plataformas já existentes que têm como propósito auxiliar os decisores nesta selecção.

A *Automated Measurement and Analysis of Open Source Software* (OSSMETER) é um projecto financiado pela UE, cuja meta é o desenvolvimento de uma plataforma (Figura 88) que apoie aqueles que tomam a decisão no processo de descoberta, comparação e monitorização da saúde, qualidade, impacto e actividade do SL/CA (http://www.ossmeter.org/, acedido em 10 Junho, 2015).

[246] No original, "*The project has designed and implemented a web application that consumes this API and enables users to register rand discover OSS projects and explore their quality indicators in a intuitive manner to aid decision making. Additionally, users are able to receive notifications when projects of interest slip below (or above) selected thresholds on the set of metrics that area most important to them*." *In* OSSMETER (2015) – **Disponível online** no endereço URL http://www.ossmeter.org/ossmeter-demo (acedido em 19 Dezembro, 2015).

Figura 88 - OSSMETER – plataforma para comparação entre *software* e aferição da validade do processo de adoptação. Fonte: http://www.ossmeter.org/ossmeter-demo (acedido em 10 Junho, 2015).

Segundo as informações disponibilizadas na página oficial do projecto, esta aplicação *web* providencia uma API para pesquisa de dados e sua integração em aplicações de outros desenvolvedores. Entre estes dados constam informações sobre os projectos de SL/CA, e indicadores de qualidade sobre eles. Infelizmente, o projecto encontra-se ainda em *prototype stage*, esperando-se a saída de um demo *online* o mais brevemente possível.

Em Portugal, como se teve oportunidade de referir, o Decreto 107/2012 atribui à AMA a responsabilidade e publicitar as normas e avaliação e a metodologia (cálculo do TCO) que os órgãos da APP devem utilizar para comparar soluções. Os administradores que têm a seu cargo esta tarefa podem descarregar, no sítio da AMA, o formulário de pedido de parecer para autorização da aquisição de *software* (folha de cálculo) previamente programado para determinar a solução mais económica (Figura 89). No entanto, na nossa óptica, considera-se que esta aplicação é insuficiente, por duas razões fundamentais:

i. O documento proposto pela AMA considera uma dimensão de Risco (organizacional, tecnológico e de instalação/implementação), que diz respeito à integração de indicadores relacionados com as características do pessoal ao serviço e à eficiência tecnológica da solução informática. No entanto, a estimação deste tipo de factor é incipiente e genérica,

pois não constitui uma averiguação rigorosa da eficiência do *software* ou dos impactos do projecto na estrutura organizacional, visto que se resume a um conjunto de perguntas gerais que não auferem informação relevante sobre a verdadeira capacidade de trabalho com aquele *software* ou sobre medidas de *benchmarking*. Dá-se primazia ao valor económico quantificado em euros, tal como se visualiza na Figura 89[247]. Logo, variáveis relacionadas com os aspectos técnicos e organizacionais devem merecer uma análise mais exaustiva. Conclui-se que esta forma de cálculo não é suficiente[248];

ii. Claro que, tendo o foco naqueles que têm competências na área das TIG, julgamos que a APP carece de uma interface dedicada a este tipo de TI que englobe parâmetros técnicos, à semelhança do *GIS Software Selection Advisory System* (Eldrandaly & Naguib, 2013), e que integre também outros tipos de custo (monetários, organizacionais e outros).

Figura 89 - Formulário para pedido de parecer à AMA para aquisição de *software* – Análise do custo de *software* ou TCO. Fonte: https://m6.ama.pt/ (acedido em 13 Novembro, 2014).

[247] A TCO pode ser vista de duas formas. Por um lado, se a considerarmos apenas como o custo monetário associado à aquisição/implementação, a médio/longo prazo, de um programa, a TCO é uma medida insuficiente, uma vez que ignora todo um conjunto de factores preponderantes, na linha do que se tem dito. Por outro lado, se incluirmos na TCO todos os custos associados ao *software*, quer eles sejam monetários ou não, aproximamo-nos da realidade. Assumimos a nossa clara preferência por este segundo ponto de vista.

[248] No formulário, a avaliação dos recursos humanos é reduzida a quatro perguntas; uma delas é´: "Qual a proporção de elementos da equipa com experiência na tecnologia e/ou em projectos semelhantes?" Sem dúvida, esta é uma questão válida, ainda assim insuficiente. Por exemplo, se a resposta nos disser que existe uma grande quantidade de elementos capazes, o Risco é menor; se a resposta for no sentido contrário, o Risco é certamente maior, contudo não nos podemos contentar com esta resposta, importa auscultar o pessoal, percebendo qual é a sua capacidade de aprendizagem de uma nova tecnologia. Assim, embora exista sempre um custo acrescido (monetário ou não) com formação, esse custo pode não ser significativo ao ponto de justificar a não adopção dessa tecnologia. Estas insuficiências estendem-se claramente à parte técnica.

No âmbito dos SIG, recordamos o sistema proposto por Eldrandaly & Naguib (2013) apresentado na Figura 40. Esta é a única plataforma do género a operar no âmbito dos SIG e é um dos pontos de partida para a construção da SIGósAPp que, ao contrário da primeira, não se preocupará apenas com as capacidades técnicas que o *software* tem para a resolução de um problema específico, incluindo outras variáveis como as referidas anteriormente.

Em suma, todos estes exemplos servem-nos de referência e constituem a matriz conceptual da SIGósAPp que consolidará numa única interface as propostas da AMA e de Eldrandaly & Naguib (2013), contemplando também outros aspectos relacionados com a dimensão psico-organizacional do problema.

4. Síntese: primórdios da formulação de um MDS4OSS com enfoque no *Software* SIG *Desktop*

> *"De acordo com a nossa experiência de uso de SL/CA, e motivados pelas nossas responsabilidades, devemos mencionar que há pouco conhecimento partilhado sobre a utilização deste tipo de software. Assim, é necessário proceder a uma forte avaliação qualitativa e quantitativa de SL/CA em geral, e, em particular, de SL/CA SIG."*
>
> Sanchez *et al.* (2007, *apud* Akabari & Rajabi, 2013:757)[249]

Existe margem para a formulação de uma plataforma para efectuar um balanço comparado entre soluções informáticas SIG? Cremos que sim. Em primeiro lugar, porque se precisa de trabalhos que acrescentem informação de qualidade sobre SL/CA e sobre a sua adopção em organizações. Em segundo lugar, esta falta de informação justifica a inexistência de um MDS4OSS especializado e aplicado a contextos reais e concretos. O projecto OSEPA contribui com linhas de rumo importantes, que indicam o caminho nos meandros do processo de migração mas, ainda assim são insuficientes, visto que se trata de guias generalistas. Assim, as dificuldades sentidas actualmente pelos administradores públicos em escolher as soluções informáticas mais sustentáveis mantêm-se e perduram.

Face ao exposto, reconhece-se o mérito destes investigadores (destes e de todos os outros que investigam a adopção de SL/CA), tal é a dificuldade do desiderato a que se propuseram, mas do qual nos desvinculamos, por julgarmos ser impossível, por agora, desenvolver algo específico e dedicado que se distinga muito do que já foi feito. Assim, tendo em conta a literatura disponível e os dados de que dispomos, tratamos esta rubrica da dissertação como uma primeira avaliação das

[249] No original, *"According to our experience with using free/open source software, and based on our responsibilities, we should mention that there is a shortage of knowledge and information in using this software. There is a need for a strong evaluation of FOSS in general and in particular free/open source GIS software". In* Sanchez *et al. apud* AKBARI, Mohammad & RAJABI, Mohammad (2013) – "Evaluation of desktop free/open source GIS *software* based on functional and non-functional capabilities". **Tehnički vjesnik**, Vol. 20, N.º 5, 755-764.

condições das quais não podemos prescindir quando nos propusermos a apresentar um MDS4OSS com enfoque nos SIG.

Esta constatação serve também para a formulação da interface SIGósAPp que visa a selecção automática do *software* economicamente mais vantajoso para uma determinada organização pública, que envolva todos os factores e dimensões (estipulação de um modelo de adopção – Figura 90-A –, e desenvolvimento de um modelo para selecção de *software* – Figura 90-B) que, segundo os autores mencionados, se acharem pertinentes e parametrizáveis, tendo por base a Teoria da Difusão e Adopção da Inovação e as premissas relacionadas com a boa gestão da mudança. É com algum constrangimento que, chegados a esta fase do trabalho reconhecemos que no momento presente, ainda não dispomos de recursos (tempo e conhecimentos) para a apresentação do primeiro protótipo da SIGósAPp, sobretudo porque as referências disponíveis não nos indicam os melhores métodos para valoração dos parâmetros envolvidos no balanço TCO quando consideramos dimensões que transcendem a financeira. Ainda assim, não deixaremos de adiantar as primeiras considerações sobre a sua conceptualização e viabilidade, a desenvolver em trabalho futuro.

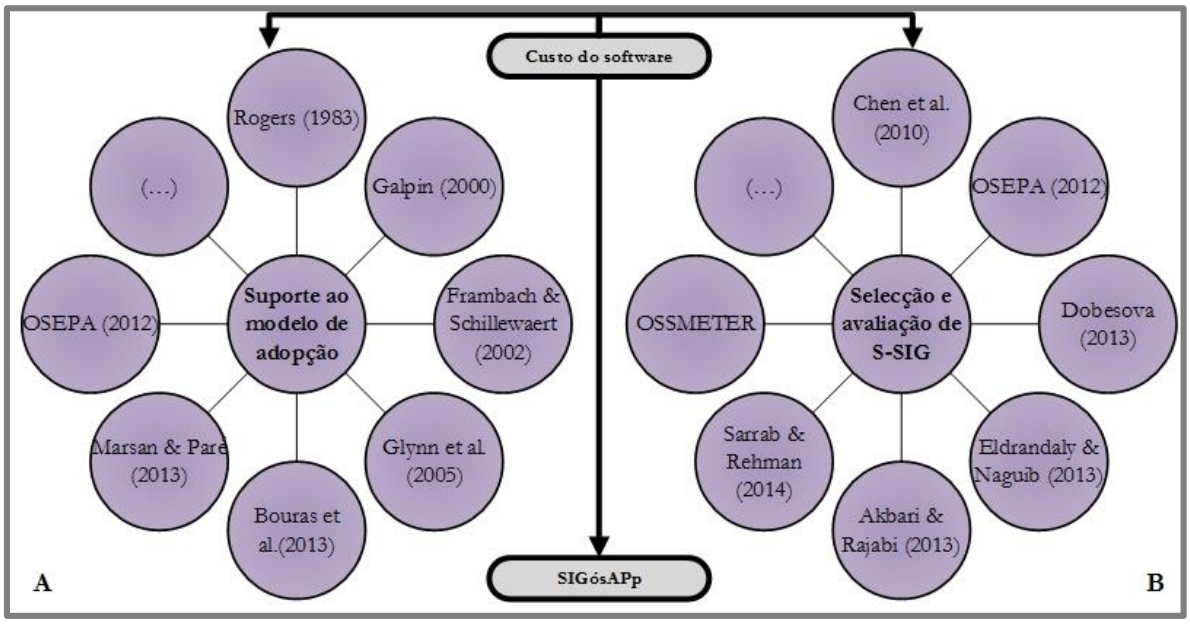


Figura 90 - Síntese do espectro bibliográfico de suporte a esta investigação em termos do apoio à constituição de um modelo de adopção para cálculo automático do custo do *software* (A) e das metodologias para seleccionar e avaliar S-SIG (B).

CAPÍTULO II – ANATOMIA E ORGÂNICA DA SIGÓSAPP – VIABILIDADE, DESENVOLVIMENTO E MANUTENÇÃO

> *"Devemos lembrar-nos que todos os modelos estão errados: a questão prática é saber quão errados é que eles têm de estar para deixarem de ser úteis. (…) Essencialmente, todos os modelos estão errados, mas alguns são úteis."*
>
> George Edward Box (1987) *apud* Rocha (2012:531)[250]

O ser humano depara-se, de forma recorrente com fenómenos/acontecimentos/contextos ou dinâmicas que o confronta com problemas que não consegue explicar ou resolver. A Ciência é, antes de mais, um modo de lidar com aquilo que não se conhece, é a estratégia concebida pelo Homem para escapar da incerteza, uma forma de viver o desconhecido descobrindo. Consequentemente, a Ciência, em qualquer área do conhecimento, busca a explicação para o desconhecido recorrendo a modelos.

Um mapa, enquanto representação gráfica e simbólica de toda ou parte da superfície da Terra e dos seus fenómenos, é um bom constructo daquilo que é um modelo. Todo o mapa, ao comunicar, mente, uma vez que resulta das deformações das propriedades geométricas da superfície (fruto da sua complexidade) e da generalização temática que torna a informação compreensível segundo os princípios da semiologia gráfica. Neste sentido, uma carta resulta de um modelo matemático que reduz a complexidade da superfície até um ponto em que esta se torna cartografável. Isto, porém, não reduz a utilidade ou a importância que as sociedades atribuem a este instrumento.

Assim, um modelo é, em última instância, a concepção teórica e humana da realidade adaptada ao entendimento de fenómenos e à resolução de problemas através da representação de características significantes ou relações, argumento simplificado (abstracção) porque, adoptando uma atitude altamente selectiva, canaliza apenas os factores, variáveis ou parâmetros que se julgam relacionados com o fenómeno, numa óptica em que simplificação significa intrinsecamente e inexoravelmente incerteza (Ferreira, 2012; Rocha, 2012)[251].

---

[250] Pensamento original apresentado na obra: BOX, George & DRAPER, Norman (1987) – **Empirical Model-Building and Response Surfaces**. John Wiley & Sons, 669 p.

[251] Seguindo esta perspectiva, o espectro daquilo a que se chama modelo inicia-se na mais simples fórmula matemática que determina o raio da circunferência e termina na maior sequência de processos que tenham como propósito

O modelo responde a uma necessidade exclusivamente humana, a do conhecimento e da resolução de problemas. Os sistemas não existem na natureza ou nos meios humanos, apenas no nosso pensamento, por isso, só a decomposição da realidade em camadas permite perceber o modo de funcionamento do topo. Face ao exposto, numa perspectiva holística, os modelos não são simples representações acerca do mundo, mas sim formulações do nosso conhecimento acerca do mundo, isto é, por um lado, o processo de modelação está sempre dependente da forma de raciocínio de quem o pensou, por outro, não são fonte de conhecimento absoluto, são somente projecções do que irá acontecer, predições, simulações, aproximações, pois não deixam de ser uma simplificação (Ferreira, 2012; Rocha, 2012).[252]

Esta simplificação, esta escolha das estruturas da realidade que se consideram essenciais para o objectivo do operador, traduz-se num enviesamento conceptual, da mesma forma que, por vezes, não existem formas lógicas ou matemáticas capazes de representar uma destas estruturas tal como elas existem na realidade e, como tal, todos os modelos têm intrinsecamente associado um erro, que só o é porque não é possível decifrá-lo com exactidão. Todavia, apesar da incerteza, os modelos são a melhor forma de entender o funcionamento, cada vez mais complexo do que nos rodeia (Ferreira, 2012).

Neste contexto, destacam-se quatro níveis de abstracção (níveis de generalização ou simplificação) – Figura 91. No primeiro nível temos a realidade que é constituída por fenómenos reais e inclui aspectos que podem ser ou não percebidos pelos indivíduos. Em segundo lugar, consoante a interpretação individual dos fenómenos reais, gera-se um raciocínio estruturado que selecciona os processos e objectos que se consideram relevantes para a resolução de um problema específico, tornando-o, por isso, independente dos detalhes da implementação. Com base neste último, formula-se o modelo lógico, que especifica como implementar o anterior, ou seja, que estabelece a estrutura do sistema, tendo em linha de conta os métodos técnicos, informáticos ou de experimentação a serem utilizados. Por fim, o modelo físico é o resultado do processo de programação e implementação do sistema.

encontrar vida noutros planetas. De facto, um modelo pode ser uma teoria, uma lei, uma hipótese ou uma ideia minimamente estruturada, uma equação, uma representação de um sistema para estudar uma parte ou para o estudar como um todo (Rocha, 2012).

[252] Como bem elucida Ferreira (2012), o termo 'simplificação' não deve ser apreendido como um entendimento simplista da realidade, mas antes como a estruturação da realidade de maneira a salientar apenas os elementos que se julgam serem interessantes para descrever, compreender e explicar o fenómeno em causa.

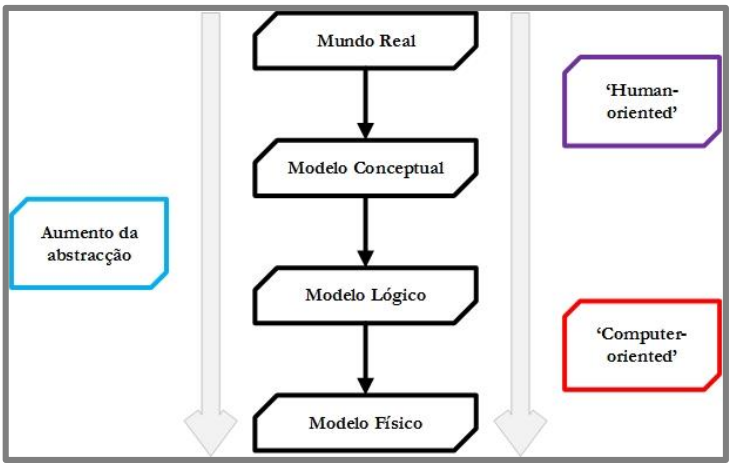


Figura 91 - Níveis relevantes de abstracção em modelação. Fonte: Longley *et al.* (2005), p. 179.

A invocação destas etapas tem como propósito estabelecer uma linha de raciocínio que nos transporta desde a formulação conceptual de um modelo de cálculo de custo de uma solução informática que opera no âmbito das TIG, até à sua implementação prática numa qualquer linguagem de programação. Neste capítulo, estamos em condições de apresentar apenas o primeiro nível de generalização que diz respeito à formulação do modelo conceptual da SIGósAPp:

1. **Guia para adopção de qualquer solução tecnológica – 1ª Fase**
   i. Estabelecer o conjunto de etapas que definem o modelo/guia de adopção, no qual a execução da SIGósAPp é somente uma etapa intermédia em todo o processo.
   ii. Definir genericamente o modelo conceptual da SIGósAPp.
2. **Técnicas que implementam os fundamentos conceptuais da SIGósAPp – uma primeira abordagem ao seu modelo conceptual**
   i. Especificar o modelo conceptual da aplicação SIGósAPp, estabelecendo a relação com o conjunto de técnicas que visam a ajustada valoração dos parâmetros envolvidos no processo de decisão por uma das alternativas informáticas.

I. Guia para adopção de qualquer solução tecnológica – 1ª Fase

> "*O maior número de pesquisas relacionadas com adopção de SL/CA segue um paradigma dominante. Este paradigma procura explicar a adopção da inovação através de modelos economicistas e racionalistas… No entanto, este paradigma chegou a um 'ponto de retornos decrescentes' em termos da criação de oportunidades para pesquisas inovadoras e úteis, logo, são necessárias outras abordagens fora deste paradigma… nomeadamente as que se preocupam com factores e processos com o ambiente social das organizações que adoptam a inovação.*"
>
> Josianne Marsan, Guy Paré & Anne Beaudry (2012:259)[253]

Rogers (1983), no âmbito da Teoria da Adopção da Inovação, distingue cinco etapas, das quais a formulação da SIGósAPp não se descontextualiza, pois um organismo tem de passar por elas até à implementação de uma inovação tecnológica:

i. Conhecimento;

ii. Persuasão;

iii. Decisão;

iv. Implementação;

v. Confirmação.

Em primeiro lugar, os administradores públicos tomam contacto com a existência de alternativas ao SP com as imposições legislativas que os obrigam a solicitar à AMA um procedimento formal que os obriga a comparar e pesar ambas as soluções (SP e SL/CA), definindo e comunicando,

[253] Tradução livre a partir do original, "*The majority of prior research on IT innovation adoption has been done within what Fichman (2004) calls the dominant paradigm. According to this author, this paradigm is 'typified by the desire to explain innovation using economic-rational-istic models…It is however reaching the "point of diminishing returns" in providing additional opportunities for highly influential research, and, hence, future work stepping outside this paradigm is required… Wang (2009) introduced two theoretical perspectives that are concerned with the factors and processes in the social environment of the focal innovating organizations*. *In* MARSAN, Joisianne; PARÉ, Guy & BEAUDRY, Anne (2012) – "Adoption of open source *software* in organizations: A socio-cognitive perspective. **Journal of Strategic Information Systems**, N.º 21, 257-273.

dentro do prazo legal de 30 dias, ao decisor o resultado deste exercício. Segue-se a persuasão, que antecede a decisão, pois antes de decidir temos apresentar razões válidas que façam tombar o pêndulo para uma das alternativas em ponderação. Esta é a etapa chave de todo o processo, uma vez que garante a tomada de uma boa decisão, o que, por sua vez, torna a implementação muito mais fácil. Neste sentido, a persuasão não pode acatar interesses pessoais ou arbitrários, pelo contrário, é um processo que exige maturação e organização, tendo de responder a um conjunto de estádios fundamentais.

De acordo com a Figura 92, para persuadir os decisores na opção por uma das alternativas consideradas, devem-se cumprir as seguintes etapas:

i. Formar uma equipa que envolva impreterivelmente múltiplas perspectivas e que reúna todos aqueles (ou seus representantes) que vão ser afectados pela migração;

ii. Estabelecer metas a que a tecnologia tem de responder – definir o(s) problema(s), os objectivos do(s) projecto(s), e as propriedades/funcionalidades que a tecnologia tem de ter para satisfazer todas as necessidade afectas ao(s) problema(s) e projecto(s);

iii. Inventariar e limitar tecnologias que, numa primeira análise (com base em bibliografia e experiências passadas), têm capacidade para solucionar o problema;

iv. Testar a eficiência e qualidade dos *software* tendo por base o conjunto de especificações que ele deve cumprir;

v. Calcular, a curto e a longo prazo, os custos que se terão com aquisição/renovação de licenças, formação e suporte;

vi. Em paralelo, caracterizar a organização ou equipa que integra o projecto, tendo em vista a obtenção de um valor ou juízo que indique o custo humano da mudança;

vii. Estabelece redes, quantificar a qualidade do suporte, retirar aprendizagens de outros casos de sucesso ou insucesso.

Uma plataforma como a SIGósAPp entrará a meio do processo como uma plataforma integrada para cálculo automático de custos. O estabelecimento das metas, objectivos e especificações, a caracterização da organização e das redes externas, e o inventário de soluções mais promissoras constituem os dados de entrada, que a aplicação utilizará para valorar e ponderar correctamente as diversas variáveis que se julga por capazes de justificar a viabilidade da implementação de um determinado programa informático. Estas variáveis organizam-se, basicamente, em quatro domínios que preenchem o eixo da decisão:

i. Factores técnicos/tecnológicos;

ii. Factores financeiros relacionados com despesas monetárias;

iii. Factores sócio-humanos/organizacionais;

iv. Factores relacionados com o contexto de apoio externo.

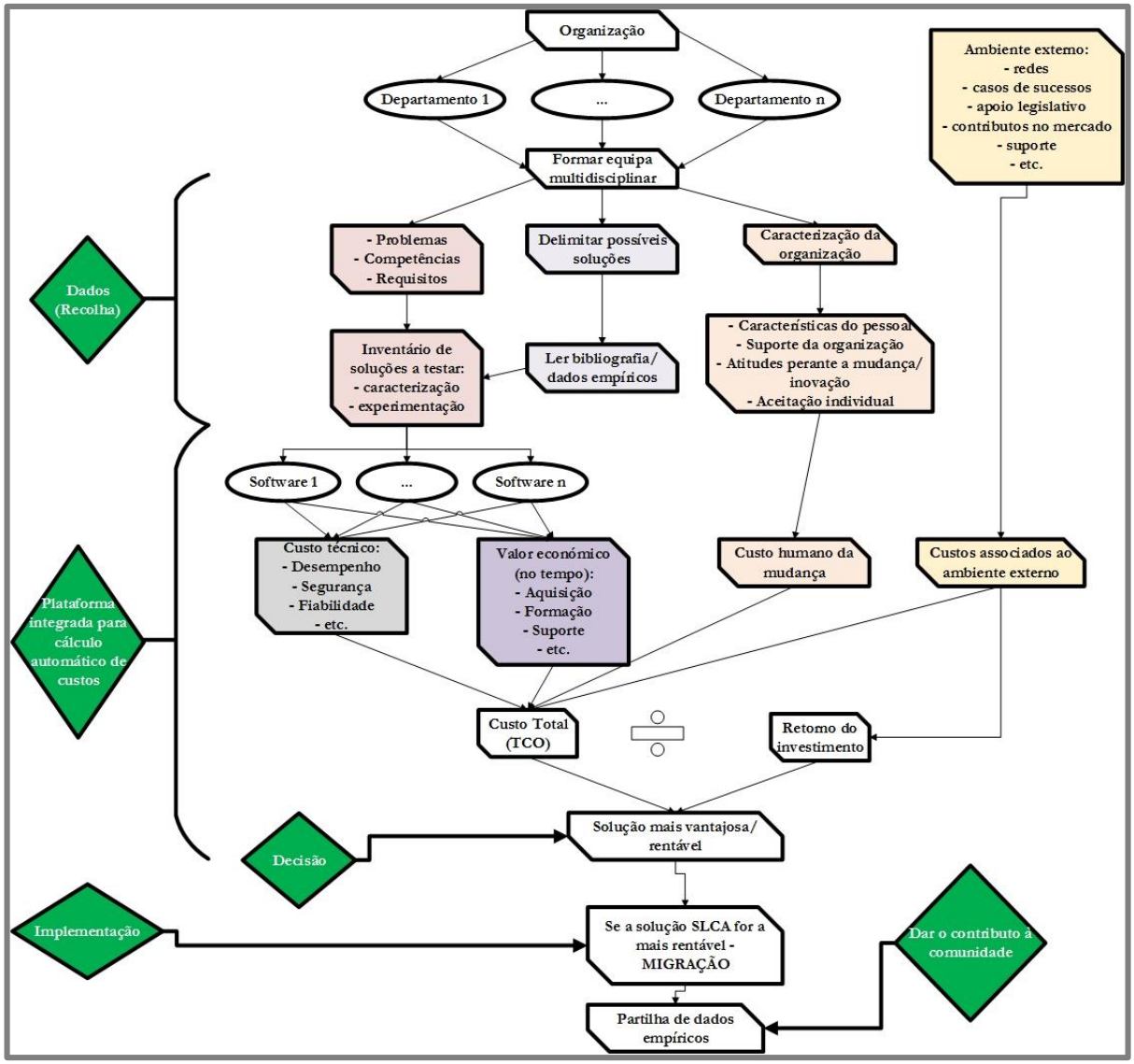


Figura 92 – Guia para adopção de qualquer solução tecnológica – 1ª Fase -, tendo em vista a tomada de decisão. Inspirado em Glynn *et al.* (2005), Fitzgerald (2011), Bouras *et al.* (2013 e 2014), Marsan & Paré (2013), Sarrab & Rehman (2014).

Cada um destes domínios gerais engloba parâmetros específicos com indicadores numéricos que reflectem a valoração do parâmetro de modo a que valores mais elevados correspondam a maior

custo. A sua soma ponderada (segundo as preferências dos decisores) diz respeito ao custo total da solução em todo o seu ciclo de vida (TCO).

Numa primeira fase, a aplicação centrar-se-á no cálculo do TCO, mas não poderíamos deixar de fora deste modelo o ROI, visto que só a contraposição do investimento realizado (TCO) com o retorno gerado (ROI) permite perceber qual a tecnologia mais rentável e vantajosa em todos os sentidos, isto porque um TCO mais baixo pode não implicar um ROI que compense os gastos iniciais; pelo contrário, um TCO mais elevado, em alguns casos pode, apesar de tudo, ser compensatório se o ROI, pelo menos, assegurar todos os investimentos feitos.

O balanço entre TCO e ROI conclui a fase da persuasão, pois os decisores passam a ter condições para tomar uma decisão em conformidade com os seus interesses. Com a decisão tomada, segue-se a implementação, na qual se devem incluir medidas que garantam uma adesão pacífica à nova tecnologia. Finalmente, entra-se na última fase, a confirmação, momento em que se verifica a adequação da tecnologia às necessidades do organismo, e em que se deve valorizar a partilha de dados empíricos sobre todo o processo, informações fundamentais para auxiliar outros administradores públicos em processos semelhantes, facilitando a tomada de decisão e a sua conformidade com o real potencial da solução a ser implementada.

O que acabamos de dizer, faz dos parâmetros específicos, afectos a cada domínio, os elementos centrais da SIGósAPp. O problema está no facto de não haver um manual que nos indique claramente que parâmetros/variáveis devem ser consideradas e qual é a forma mais correcta de os/as valorar, nomeadamente, quando relacionados/as com o contexto organizacional. A sua definição resulta da interpretação do conhecimento tácito advindo das várias problematizações que se vão fazendo sobre adopção de SL/CA em organizações, por isso, trata-se de uma tarefa complexa que exige maturação, mormente porque cada domínio precisa de uma abordagem específica, na qual devem ser tidos em conta diferentes pontos de vista (equipa multidisciplinar). Para além disso, para definirmos factores ajustados aos diversos contextos da APP, são necessários dados mais exaustivos sobre esses mesmos contextos, algo que, como já referimos, não conseguimos obter durante o desenvolvimento da presente dissertação. Consequentemente, optamos por não apresentar os vários factores relacionados com cada domínio, deixando essa tarefa para investigações futuras.

2. Técnicas que implementam os fundamentos conceptuais da SIGósAPp – uma primeira abordagem ao seu modelo conceptual

> *"Construir um projecto SIG é um grande investimento. A escolha do software SIG correcto é uma decisão crítica para o sucesso ou insucesso desse investimento. Devido à complexidade do problema, devem ser utilizadas um conjunto de técnicas para tomada de decisão tendo em vista a chegada a uma solução adequada."*
>
> Khalid Eldrandaly & Soad Naguib (2013:152)[254]

A implementação da SIGósAPp, em termos gerais, segue o suporte lógico da AMC (Figura 93). A meta diz respeito à escolha do *software* mais eficiente e rentável para a organização nesse contexto ambiental (conceito estruturante da AMC, tal como se referiu anteriormente), contexto que pode ser mais ou menos extensível, consoante o período de vida que se pretenda dar à tecnologia. Cada um dos domínios enunciados anteriormente encarna a posição de objectivo, ao qual estão associados vários atributos, os tais parâmetros que ainda estão por definir e que estarão assentes na raiz da SIGósAPp.

[254] No original, *"Building a new GIS project is a major investment. Choosing the right GIS software package is critical to the success and failure of such investment. Because of the complexity of the problem a number of decision making tools must be deployed to arrive at the proper solution*". *In* ELDRANDALY, Khalid & NAGUIB, Soad (2013) – "A Knowledge-Based System for GIS *Software* Selection". **The International Arab Journal of Information Technology**, Vol. 10, N.º 2, 152-159.

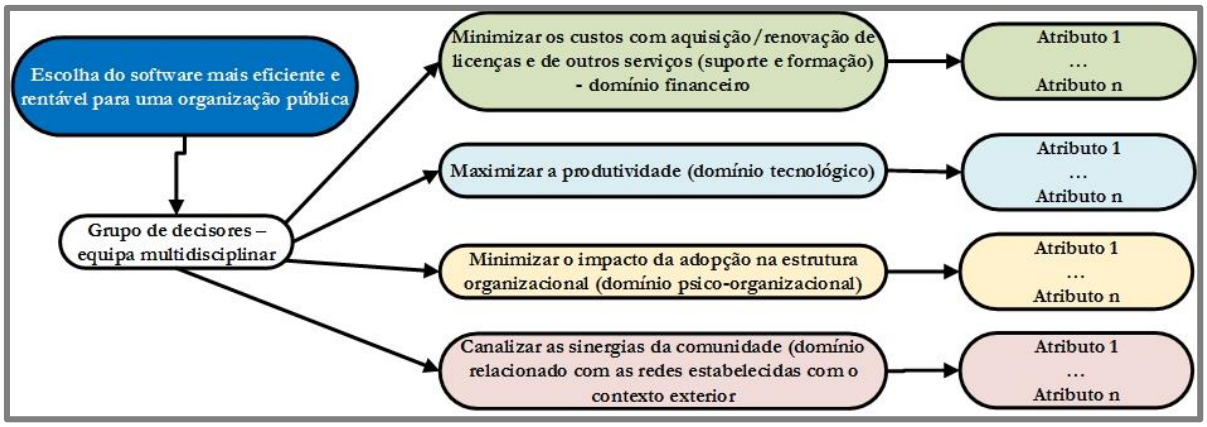


Figura 93 - Estrutura da análise multicritério a implementar para o desenvolvimento da SIGósAPp.

A valoração e a agregação dos vários atributos num único valor que represente a predisposição da alternativa para o cumprimento da meta dependerá dos dados disponíveis num formulário e numa BD, dois elementos que fazem parte da orgânica da plataforma (Figura 94), e sem os quais ela não poderá funcionar.

Na linha da Figura 92, o utilizador preencherá o formulário de acordo com os dados que recolheu na fase inicial da persuasão, fornecendo como argumentos de entrada do programa, dados sobre os objectivos do projecto, *software* que se deseja comparar e especificações que deve respeitar, características da organização, e esclarecimentos quanto aos contactos que a organização estabelece com a comunidade de SL/CA e com outras organizações que já implementaram estas soluções (casos de sucesso ou insucesso). Neste formulário, poderão também ser definidos os pesos específicos de cada um dos parâmetros e/ou domínios que a SIGósAPp considera.

Este documento terá uma organização muito específica e restrita, da qual se espera obter respostas direccionadas, uma vez que a SIGósAPp será programada para obter dados específicos em determinadas células. A partir da leitura do ficheiro, e tendo por base as informações nele descritas, a aplicação atribuirá, a cada alternativa, um valor em cada um dos atributos incluídos no modelo de adopção que a SIGósAPp contemplará, isto nos domínios relacionados com as características da organização e contexto externo.

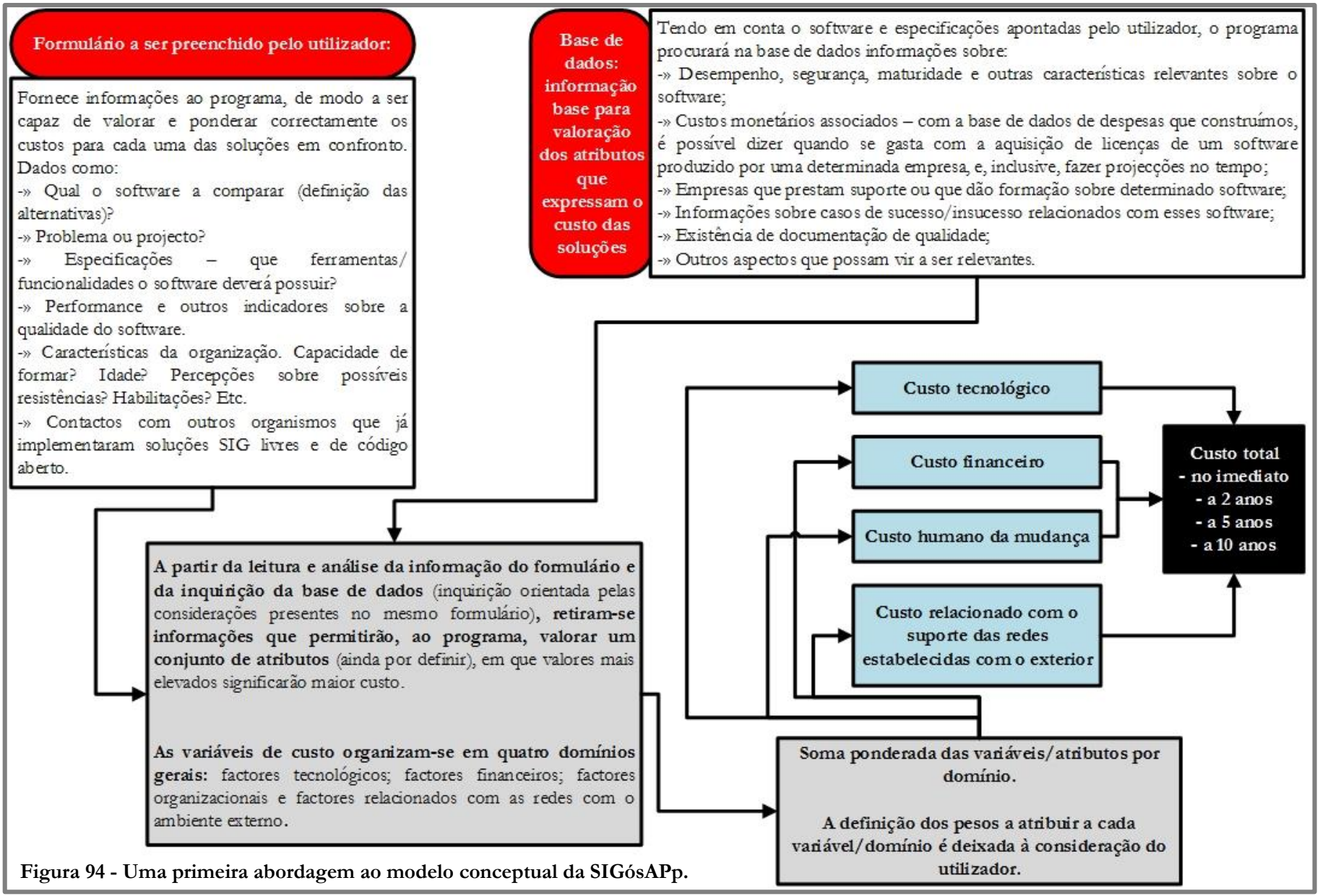


Figura 94 - Uma primeira abordagem ao modelo conceptual da SIGósAPp.

Por sua vez, os factores relacionados com a dimensão financeira e técnica serão valorados segundo os dados que constarão de uma BD que estará conectada à aplicação. Esta BD terá informações sobre os gastos associados a cada solução informática com aquisição/renovação de licenças, contratualização de serviços de suporte e de formação, e informações sobre as características das tecnologias em confronto, desde dados relativos a performance, funcionalidades, qualidade e documentação, bem como outras informações relevantes, como, por exemplo, referências a casos de sucesso ou insucesso com determinada solução SIG.

A inquirição da BD dependerá dos elementos presentes no formulário, no qual, para além da definição das alternativas a comparar, o utilizador terá de discriminar as ferramentas/funcionalidades que o *software* deverá disponibilizar para responder aos objectivos do projecto no qual se vai inserir. A aplicação será programada para, consoante o que estiver descrito no formulário, procurar na BD as informações correctas, que serão posteriormente valoradas.

A Figura 95 esboça o circuito estabelecido pela SIGósAPp desde a leitura do formulário até à obtenção de um resultado que diz respeito à hierarquização das alternativas em termos do seu custo numa perspectiva temporal (a curto, a médio e a longo prazo). Esta listagem é o resultado da soma ponderada entre todos os atributos valorados, segundo os pesos previamente definidos no formulário.

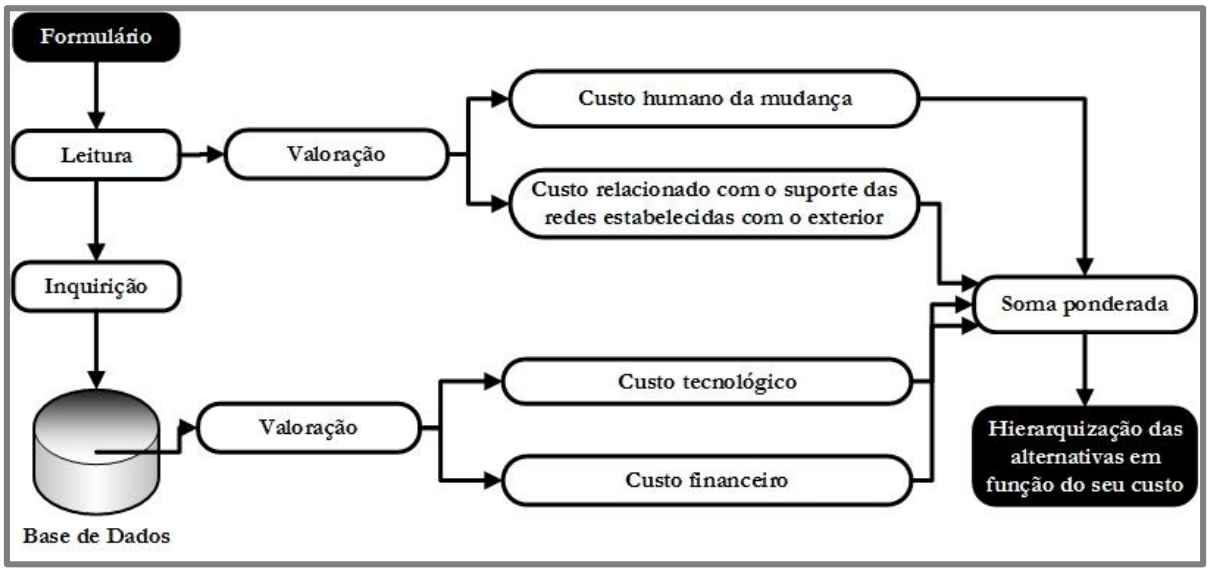


Figura 95 - Circuito lógico realizado pela SIGósAPp desde a definição dos dados de entrada até à apresentação do resultado pretendido.

> "*Um trabalho científico é, por natureza, um processo inacabado. Nasce e desenvolve-se num contexto próprio, com balizas flexíveis, em constante redefinição, no Espaço e no Tempo. Mas um trabalho de investigação é, também, fruto da relação dialéctica entre o 'saber' e o 'saber-fazer'. Os processos que conduzem ao aparecimento de novas ferramentas de trabalho e os novos estudos que convidam à actualização permanente da informação, em continuum, permitem estabelecer uma analogia com a "Teia de Penélope" e eleger a negação da "verdade absoluta" como a única das certezas que nos reservamos ter, na concepção de um processo de investigação científica."*
>
> José Gomes dos Santos (2005:489)[255]

Mais do que 'verdades absolutas' e contribuições efectivas, com o desenvolvimento desta dissertação, lançámos reflexões importantes, que procuram tornar um pouco mais transparente as matérias relacionadas com a selecção e aquisição/implementação de *software*, tendo em vista uma tomada de decisão que recaia sobre a solução mais rentável e eficiente. Neste sentido, julgamos que este trabalho, para além de inovador, apresenta conteúdos que, no pormenor, respondem essencialmente a interesses de um grupo específico de utilizadores, mas cuja utilidade se estende a toda a comunidade/sociedade, pese embora o facto de ser ainda uma primeira investigação, que deixa muito em aberto para trabalho futuro.

Logo numa primeira fase, tomámos como preocupação principal a prestação, de modo imparcial, de esclarecimentos sobre o que é o SL/CA, problematizando as suas principais vantagens e desvantagens e definindo-o como um modelo de negócio apoiado na prestação de serviços (suporte, formação, etc.), que visa, à semelhança do SP, a obtenção de lucro. Concluímos, portanto, que a designação "*software* comercial" se aplica tanto ao SP como ao SL/CA, pois não existem soluções isentas de custo.

[255] *In* SANTOS, José Gomes (2005) – *As bacias de Mirandela, Macedo de Cavaleiros e de Vilariça – Longroiva – Estudo de Geomorfologia*. **Dissertação de Doutoramento,** Faculdade de Letras da Universidade de Coimbra, 539 p.

Nestes termos, e tendo tomado conhecimento das avultadas de despesas com aquisição de licenças de SP num contexto de crise económica em que se introduziram estratégias legais que visam a optimização dos recursos e dinheiros públicos, parece-nos óbvio que a APP carece um modelo mais completo e eficaz que permita aos administradores públicos decidir em conformidade com as suas necessidades e recursos disponíveis.

Em termos estritamente financeiros, e apesar de, dependendo dos contextos, o SL/CA SIG poder ser até a alternativa mais dispendiosa, julgamos que, a médio/longo prazo, o SL/CA é a solução mais vantajosa e rentável, isto se os seus desenvolvedores assegurarem suporte e actualizações durante um período de tempo aceitável, maximizando o período de vida da tecnologia. Todavia, esta constatação não é linear, visto que os custos dependem de variáveis que transcendem a dimensão financeira. Mas importa considerar outros aspectos, tais como as características do problema que se pretende resolver com a tecnologia, a variedade e características dos *software* disponíveis capazes para resolver esse problema, a estabilidade das soluções que podem tornar-se pouco apetecíveis no caso de ao longo de um ano uma determinada aplicação apresentar três, quatro ou mais versões (o utilizador sente-se obrigado a constantes e desagradáveis adaptações), bem como o contexto organizacional em que se adoptará a inovação (migração para uma nova solução). Para além disso, como ficou bem patente, mesmo recorrendo a técnicas de *software testing* (*black-box* ou *white-box*), não é possível avaliar global e inequivocamente um *software*, no entanto, a sua avaliação em contextos particulares é possível e altamente recomendável, uma vez que os diferentes *software* apresentam distintos níveis de maturidade, na mesma medida, existe a possibilidade do mesmo *software* evidenciar comportamentos antagónicos perante dois problemas diferentes[256]. Assim, tendo em conta as imposições legislativas, a necessidade de redução da despesa e optimização de recursos, as vantagens relacionadas com a interoperabilidade, o suporte jurídico, institucional e técnico, que mostram que o SL/CA é uma solução credível, e sabendo que cada caso é um caso, concluímos que faz sentido que exista uma plataforma preparada para quantificar, de modo comparado, os custos de determinadas soluções SIG, tornando o processo de tomada de decisão mais célere e automatizado, e auxiliando de modo muito expressivo os administradores públicos nesta árdua tarefa.

O desenvolvimento futuro da plataforma SIGósAPp permitirá dar seguimento ao pontapé de saúde protagonizado por este trabalho, pois, na parte relativa ao confronto entre diversas soluções SIG, debruçámo-nos em exclusivo sobre parâmetros estritamente técnicos. Os vários exercícios

[256] Foi isso que aconteceu com o GRASS, em algumas tarefas revelou-se mais lento que a alternativa proprietária, mas, na execução de uma ferramenta específica (*r.cost*), conseguiu ser mais lesto que a concorrência.

concretizados permitiram comprovar que, no pormenor, é frequente encontrarmos soluções com diferentes níveis de maturidade o que, como se disse, acentua a necessidade de uma rigorosa avaliação técnica que denuncie as alternativas com um nível aceitável de funcionalidades, performance, desempenho, fiabilidade e qualidade.

Tínhamos como um dos objectivos nucleares comprovar que o SL/CA SIG é uma solução credível e robusta na resolução de problemas espaciais específicos enquadrados nas competências de alguns órgãos da APP e, nessa medida, este trabalho configura, por si só, uma verdadeira experiência empírica, que permitiu demonstrar a existência de alguns condicionalismos e limitações reveladas pelas alternativas em confronto. Na realização dos vários exercícios, inovámos. Por um lado, porque este é o primeiro dos estudos com carácter científico que, em Portugal, aborda o comportamento de um *software* ao longo de processos de modelação completos e relativamente elaborados (não compara apenas ferramentas de modo isolado); por outro lado, colocamos "o dedo numa ferida muito melindrosa" ao revelar a existência de discrepâncias entre *outputs* gerados por diferentes *software*, recorrendo a ferramentas que aplicam exactamente os mesmos métodos matemáticos, mas também, porque, sem fundamentalismos o fizemos, apresentámos algumas limitações do SL/CA SIG, e sem reservas demonstrámos que "nem tudo o que luz é ouro", ou seja, também o SL/CA está imbuído de um certo espírito comercial associado a modelos de negócio mais ou menos visíveis mas nem sempre perceptíveis.

Tendo em conta os problemas específicos analisados, julgamos ter sido capazes de organizar um conjunto de critérios que, segundo as nossas preferências enquanto agentes decisores, nos permitiram eleger um conjunto restrito de *software*, que experimentámos tendo em vista a avaliação da sua performance e nível de desempenho. Precisamente por se tratar de exercícios de índole técnica, a interpretação dos resultados é relativa ao peso que um determinado agente decisor atribui a este domínio[257]. Consequentemente, não argumentaremos que as diferenças de qualidade/maturidade entre SP SIG e SL/CA SIG são suficientes (ou não) para inviabilizar a migração para estas soluções, ao invés, deixamos essas considerações para a interpretação e juízo individual de cada grupo de administradores públicos.

Esta é a postura que nos parece ser mais correcta uma vez que, perante exercícios tão específicos, não nos é possível, neste ponto, fazer generalizações sobre performances ou quaisquer outros

[257] Nos organismos em que se dá primazia à qualidade técnica em detrimento das outras dimensões/domínios, estes resultados têm um determinado peso; pelo contrário, em organismos em que a redução de custos (em euros) é absolutamente incontornável, existindo também pessoal ao serviço totalmente habilitado e capacitado para operador soluções livres e de código aberto, provavelmente as diferenças de desempenho e os pontuais condicionalismos que apresentámos têm um peso necessariamente diferente, relativamente à hipótese anterior.

parâmetros de qualidade. O problema da realização de testes específicos é que a maior parte dos organismos tem outros problemas para resolver, inclusive, alguns extravasam a modelação espacial e a geoestatística, exigindo outro tipo de abordagem metodológica, como por exemplo, o armazenamento e disponibilização de informação geográfica. Face ao exposto, as soluções que tiveram mais dificuldades para processar os dados de entrada dos nossos ensaios, embora aparentem apresentar algumas limitações em tarefas de modelação espacial e geoestatística, assumem outras valências, que justificam a existência de casos de sucesso, uns documentados, outros ainda estão por documentar.

Ora, entre as soluções testadas, o GRASS GIS parece ser a alternativa mais segura quando o objectivo passa pela aplicação de técnicas de modelação espacial e de análise de imagens de satélite, concorrendo, neste campo específico, com a OrfeoToolbox e com o *software desktop* Monteverdi, as grandes referências de código aberto para processamento de dados no âmbito da DR. Isto é comprovado pela extensa lista de aplicações em diversas áreas do conhecimento (https://grass.osgeo.org/documentation/applications/, acedido em 23 Dezembro, 2015), consequência do facto de o GRASS GIS possuir uma extensa biblioteca de algoritmos, em nada inferior à do ArcGIS.

Por sua vez, o QGIS não é tão adequado para processamento geoestatístico e análise avançada de dados espaciais, pois possui poucas ferramentas nativas e mostra-se pouco eficiente no processamento de um grande volume de dados. Contudo, tal constatação não invalida que seja uma alternativa bastante forte quando o objectivo é a manipulação, armazenamento e edição básica de informação geográfica, uma vez que possibilita, de um modo mais intuitivo que os sistemas proprietários, a integração e comunicação entre um SIG *desktop* e o SGBD PostgreSQL/PostGIS, na mesma medida que permite fácil acesso a dados que estão disponíveis remotamente através da sua integração com outros aplicativos da família QGIS (QGIS Server e QGIS *Cloud*) e outros serviços de *Webmapping* e *WebSIG*. Na realidade, os resultados obtidos e a pesquisa bibliográfica que realizámos levam-nos a admitir que, no âmbito das TIG, o SL/CA é mais utilizado em aplicações que implementam sistemas SIG na *web* (*WebSIG*), sendo menos frequentes as utilizações em fins relacionados com modelação espacial.

Por fim, no mundo do SL/CA SIG, a biblioteca GDAL/OGR assume um papel fundamental, o de assegurar a interoperabilidade entre formatos e plataformas, de tal modo que é capaz de ler e realizar conversões entre um conjunto de 226 formatos (tanto em modelo de dados *raster* como em vectorial). Como consequência, esta biblioteca faz parte das distribuições de código aberto mais relevantes. No entanto, ao focar-se nestas questões, dispõe de um conjunto ainda pouco amplo de

ferramentas de geoprocessamento, ao mesmo tempo que exige do operador conhecimentos avançados em programação.

Portanto, ao desaconselharmos generalizações e relativizando o valor dos resultados, assumimos que o verdadeiro contributo do trabalho é a problematização de métodos e técnicas (AMC e PAH), cuja implementação tem como propósito matricial auxiliar no processo de decisão. Ainda assim, sobretudo para a comunidade de desenvolvedores, não deixa de ser importante tomar contacto com as limitações funcionais e de performance que algum SL/CA ainda revela, potenciando a resolução de alguns dos problemas enunciados *(bug reporting*). Esta é a essência do SL/CA, a detecção de erros potencia a sua resolução e o exponencial desenvolvimento e melhoria do *software*… para todos!

Recordando as palavras de DiBona (2006), o SL/CA não é mágico, não trabalha à velocidade da luz, muitas vezes as equipas são pequenas e os *bugs* acontecem, de tal modo que nos cabe a nós a discriminação e resolução contínua e gradual dos problemas encontrados, pois só assim, com esta melhoria gradual, a utilização do SL/CA se generalizará, algo que talvez ainda não aconteça porque as complicações que existem obrigam o utilizador a aplicar conhecimentos de programação, alterando o código-fonte, ou construindo uma solução alternativa, tal como aconteceu várias vezes ao longo deste trabalho. No âmbito dos SIG, grande parte dos utilizadores não dispõe de uma base de conhecimentos sólidos em programação e computação, o que pode motivar alguma inércia e resistência à mudança e à inovação, bem como implicará a cristalização de procedimentos que inviabilizam a adopção de SL/CA.

Por outro lado, não poderíamos terminar este trabalho sem fazer referência ao facto de ele constituir prova de que a aprendizagem de um novo *software* SIG está ao alcance de todos aqueles que são capazes de conceber cadeias lógicas de raciocínios que implementam teorias matemáticas, métricas e lógicas espaciais, tendo em vista a obtenção de um *output* para o apoio à decisão em matérias relacionadas com o espaço geográfico, nomeadamente, o Ordenamento do Território. Quando avançámos para o desenvolvimento desta dissertação, não dispúnhamos de conhecimentos sólidos e avançados em nenhuma das soluções livres e de código aberto testadas, portanto, tudo o que apresentámos é consequência de um processo de aprendizagem isolado e autónomo. No entanto, esta aprendizagem não se debruçou sobre modos de analisar os fenómenos, pois essas estruturas e técnicas, na sua dimensão conceptual, já estavam bem cimentadas no nosso intelecto, fruto dos ensinamentos assimilados ao longo da frequência no Mestrado de Tecnologias de Informação Geográfica. Portanto, todas estas aprendizagens que referimos decorrem da aquisição de capacidades e competências para manipulação de novas

interfaces, e só foram possíveis porque já possuíamos os conhecimentos que se referiram anteriormente. Enfim, conotamos esta exposição de uma dimensão pedagógica que tenta transmitir a quem inicia agora o seu estudo sobre SIG que, ao contrário do que maior parte dos aprendizes pensa, o problema não é a manipulação de um *software*, mas sim o conhecimento das estruturas espaciais e o desenvolvimento de uma capacidade de abstracção que nos permita converter esses mesmos conceitos da realidade em camadas abstractas que, ao serem combinadas entre si segundo um conjunto de pressupostos lógicos, permitem a obtenção de conclusões sobre a solução para um determinado problema da sociedade; por outras palavras, o raciocínio SIG prevalece sobre o *software* SIG e dele deve ser independente. Trabalha-se em SIG e não com o *software* "X" ou "Y", numa lógica de interoperabilidade.

Para além da desconstrução destas mentalidades menos esclarecidas que vêem o SIG apenas como um *software*/ferramenta, esperamos que as várias rotinas usadas ao longo desta dissertação possam também servir de ajuda (e de motivação) para aqueles que têm interesse em alargar os seus conhecimentos técnicos com o manuseamento de SL/CA SIG por meio de programação, veículo fulcral quando existe interesse (ou necessidade) de se automatizarem procedimentos.

Finalmente, a nossa contribuição material (se se preferir, mais tangível) com esta dissertação teria sido bem maior se tivéssemos reunido condições para consumar o desenvolvimento da plataforma SIGósAPp; todavia, pelas razões invocadas na primeira parte deste trabalho, não nos foi possível desenvolver o protótipo desta plataforma, adiando esta tarefa para trabalho futuro. No entanto, a aplicação SIGósAPp só será viável se estiverem reunidas algumas condições fundamentais:

i. A colaboração das entidades com responsabilidades nesta matéria é fulcral, na medida em que são fontes de informações sobre o enquadramento legal dos vários contextos dos órgãos da APP, ao mesmo tempo que têm força para mobilizar estes organismos, de modo a que seja possível a recolha massiva de experiências empíricas relevantes.

ii. A definição dos parâmetros/atributos que expressam o custo associado a cada domínio (tecnológico, financeiro, organizacional, e relacionado com as redes estabelecidas com o exterior), e de estratégias para a sua valoração só será acertada se estiver envolvida no projecto de construção da SIGósAPp, uma equipa multidisciplinar, que reúna conhecimentos relacionados com *software* SIG e *software testing*, economia, e psicologia das organizações.

iii. Por último, visto que o SL/CA se encontra em constante mutação e actualização, a consolidação de uma BD, que inclua informações sobre a aquisição de licenças e

contratualização de outros serviços relacionados com *software*, e informações sobre a qualidade e robustez da tecnologia, terá de ser inexoravelmente um esforço colaborativo, alargado à comunidade de desenvolvedores, visto que o confinamento desta tarefa a um grupo restrito de pessoas conduzirá à desactualização permanente da referida BD.

1. AFZAL, Wasif; ALONE, Snehal; GLOCKSIEN, Kerstin & TORKAR, Richard (2016) – "*Software* test process improvement approaches: A systematic literature review and an industrial case study". **The Journal of Systems and *Software***, N.º 111, 1-33.
2. AGTERBERG, F. & CHENG, Q. (2002) – "Conditional independence teste for weights of evidence modelling". **Natural Resources Research**, N.º 11, 249-255.
3. AKBARI, Mohammad & RAJABI, Mohammad (2013) – "Evaluation of desktop free/open source GIS *software* based on functional and non-functional capabilities". **Tehnički vjesnik**, Vol. 20, N.º 5, 755-764.
4. AOUN, Nael; MATSUDA; Hirotaka & SEKIYAMA, Makiko (2015) – "Geographical accessibility to healthcare and malnutrition in Rwanda. **Social Science & Medicine**, N.º 130, 135-145.
5. APARICIO, Manuela & COSTA, Carlos (2012) – Macroeconomics leverage trough open source. In *Proceedings of the Workshop on Open Source and Design of Communication*, Lisboa, 11 Junho 2012 (19-24).
6. APDSI (2004) – Open Source *Software* – Que oportunidades em Portugal? 46 p. Associação para a Promoção e Desenvolvimento da Sociedade da Informação. Lisboa. Portugal. **Disponível online** no endereço URL: http://www.algebrica.pt/i_ap/bo2/data/upimages/Estudo_Open_Source_com_capa.pdf (acedido em 9 Abril, 2015).
7. BAYTIYEH, Hoda & PFAFFMAN, JAY (2010) – "Open source *software*: A community of altruists". **Computers in Human Behavior**, N. º 26, 1345-1354.
8. BEGUERIA, Santiago (2006) – "Validation and evaluation of predictive models in hazard assessment and risk management". **Natural Hazards**, Vol. 37, Issue 3, 315-339.

9. BITZER, Jürgen & SCHRÖDER, Philipp (2007) – "Open Source *Software*, Competition and Innovation". **Industry and Innovation**, Vol. 14, N. º 5, 461-476.

10. BLANSIT, B. Douglas (2009) – "Free/Open Source *Software* Licenses". **Journal of Electronic Resources in Medical Libraries**, N. º 6, 362-370.

11. BONHAM-CARTER, Graeme (1994) – **Geographic Information Systems for Geoscientists: Modelling with GIS**. Pergamon Press, 1ª Edição, Oxford, 391 p.

12. BORZACCHIELLO, Maria & GRAGLIA, Massimo (2013) – "Estimating benefits of Spatial Data Infrastructures: A case study on e-Cadastres". **Computers, Environment and Urban Systems**, N.º 41, 276-288.

13. BOURAS, Christos; KOKKINOS, Vasileios & TSELIOU, Georgia (2013) – "Methodology for Public Administrators for selecting between open source and proprietary *software*". **Telematics and Informatics**, N. º 30, 100-110.

14. BOURAS, Christos; FILOPOULOS, Anestis.; KOKKINOS, Vasileios.; MICHALOPOULOS, Sotiris.; PAPADOPOULOS, Dimitris. & TSELIOU, Georgia. (2014) – "Policy recommendations for public administrators on free and open source *software* usage". **Telematics and Informatics**, N.º 31, 237-252.

15. BURROUGH, Peter & MCDONNELL, Rachael (2000) – **Principles of Geographical Information Systems**. Oxford University Press, 3ª Edição, Nova Iorque, 333 p.

16. CARILLO, Kévin (2014) – *Understanding contributor behaviour within Free/Libre/Open Source Software Communities: a socialization perspective*. **Dissertação de Doutoramento**, Victoria University of Wellington, 288 p.

17. CASTRONOVA, Anthony; GOODALL, Jonathan & ELAG, Mostafa (2013) – "Models as web services using the Open Geospatial Consortium (OGC) Web Processing Service (WPS) standard". **Environmental Modelling & *Software***, N.º 41, 72-83.

18. CHAUHAN, Chitij (2015) – **PostgreSQL Cookbook**. Packt Publishing, 1ª Edição, Birmingham, 270 p.

19. CHEN, Daoyi; SHAMS, Shahriar; CARMONA-MORENO, César & LEONE, Andrea (2010) – "Assessment of open source GIS *software* for water resources management in developing countries". **Journal of Hydro-environment Research**, N. º 4, 253-264.

20. CHEN, Nengcheng; ZHANG, Xiang & WANG, Chao (2015) – "Integrated open geospatial web service enabled cyber-physical information infrastructure for precision agriculture monitoring". **Computers and Electronics in Agriculture**, N.º 111, 78-91.

21. CHESNEY, Tanya; MONTERO, Jose; HEPPELL, Selina & GRAHAM, Jim (2013) – "Interannual variability of Humboldt squid (Dosidicus Gigas) of Oregon and Southern Washington". **Calco Fire Reports**, Vol. 54, 1-12. **Disponível online** no endereço URL: http://www.calcofi.org/publications/calcofireports/v54/Vol_54_Chesney.pdf (acedido em 28 Abril, 2015).

22. COELHO, Rui; INFANTE, Paulo & SANTOS, Miguel (2013) – "Application of Generalized Linear Models and Generalized Estimation Equations to model at-haulback mortality of blue sharks captured in a pelagic longline fishery in the Atlantic Ocean". **Fisheries Research**, N. º 145, 66-75.

23. CORDEIRO, António & GUIMARÃES, Carlos (2013) – "Áreas de prevenção de riscos naturais – a sua aplicação no contexto do planeamento, no concelho da Figueira da Foz", *in* LOURENÇO, Luciano & MATEUS, Manuel (coord.): Riscos Naturais, Antrópicos e Mistos – Homenagem ao Professor Doutor Fernando Rebelo, Coimbra, Departamento de Geografia da Faculdade de Letras da Universidade de Coimbra, 215-233.

24. COSME, António (2012) – **Projeto em Sistemas de Informação Geográfica.** LIDEL – Edições Técnicas, Lisboa, 361 p.

25. CRUDEN, D. & VARNES, D. (1996) – "Landslide Types and Processes", *in* TURNER, A. & SCHUSTER, R. (Eds.): Landslides, Investigation and Mitigation, Washington D. C., National Academy Press, 36-75.

26. DAVIES, Alex (2010) – **High Availability MySQL Cookbook**. Packt Publishing, 1ª Edição, Birmingham, 261 p.

27. DEDRICK, Jason & WEST, Joel (2004) – An Exploratory Study into Open Source Platform Adoption. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, Honolulu, 5 – Janeiro 2004, DOI: 10.1109/HICSS.2004.1265633.

28. DÍAZ, Laura; BRÖRING, Arne; MCINERNEY, Daniel; LIBERTÁ, Giorgio & FOERSTER, Theodor (2013) – "Publishing sensor observations into Geospatial Information Infrastructures: A use case in fire danger assessment". **Environmental Modelling & *Software***, N.º 48, 65-80.

29. DIBONA, Chris (2006) – "Open Source and Proprietary *Software* Development", *in* DIBONA, Chris; COOPER, Danese & STONE, Mark (eds.): Open Source 2.0 – The Continuing Evolution, Sebastopol, O'Reilly Media, 21-36.

30. DOBESOVA, Zdena (2013) – "CartoEvaluation Method for assessment of GIS *software*". **Geodesy and Cartography**, Vol. 39, N. º 4, 164-170.

31. DODGSON, Mark; GANN, David & SALTER, Ammon (2008) – **The Management of Technological Innovation – Strategy and Practice**. Oxford University Press, 1ª Edição, Nova Iorque, 373 p.

32. DORMAN, Michael (2014) – **Learning R for Geospatial Analysis**. Packt Publishing, 1ª Edição, Birmingham, 330 p.

33. EL-SHEIMY, Naser; VALEO, Caterina & HABIB, Ayman (2005) – **Digital Terrain Modelling**: **Acquisition, Manipulation, and Applications**. Artech House, 1ª Edição, Norwood, 257 p.

34. ELDRANDALY, Khalid & NAGUIB, Soad (2013) – "A Knowledge-Based System for GIS *Software* Selection". **The International Arab Journal of Information Technology**, Vol. 10, N.º 2, 152-159.

35. ENGELHARDT, Sebastian & FREYTAG, Andreas (2013) – "Institutions, culture, and open source". **Journal of Economic Behavior & Organization**, N. º 95, 90-110.

36. EPIFÂNIO, Bruno; ZÊZERE, José Luís & NEVES, Mário (2014) – "Susceptibility assessment to diferente types of landslides in the coastal cliffs of Lourinhã (Central Portugal)". **Journal of Sea Research**, N.º 93, 150-159.

37. FEDERSPIEL, Sofie & BRINCKER, Benedikte (2010) – "*Software* as Risk: Introduction of Open Standards in the Danish Public Sector". **The Information Society**, N. º 26, 38-47.

38. FERNANDES, Nuno (2010) – *O software open source como suporte à infra-estrutura de TI na administração pública local portuguesa: Factores determinantes na sua adopção*. **Dissertação de Mestrado**, ISCTE – Instituto Universitário de Lisboa, 177 p.

39. FERNÁNDEZ, M.; HAMILTON, H. & KUEPPERS, L. (2013) – "Characterizing uncertainty in species distribution models derived from interpolated weather station data". **Ecosphere**, Vol. 4, Issue 5, 1-17. **Disponível online** no endereço URL: http://www.esajournals.org/doi/pdf/10.1890/ES13-00049.1 (acedido em 28 Abril, 2015).

40. FERREIRA, Rui (2001) – *Modelação cartográfica em ambiente SIG para apoio à decisão: aplicação da afectação potencial de usos do solo no sector Norte do Maciço Marginal de Coimbra*. **Dissertação de Mestrado**, Faculdade de Letras da Universidade de Coimbra.

41. FERREIRA, Rui (2012) – *Estrutura da Paisagem e Modelação da Ocupação do Solo – Aplicação aos concelhos de Aveiro, Viseu e Guarda*. **Dissertação de Doutoramento**, Faculdade de Letras da Universidade de Coimbra, 425 p.

42. FERREIRA, Rui (2013) – "Acessibilidade rodoviária na área de Coimbra. A propósito de um mapa de isócronas publicado por Fernando Rebelo em 1975", *in* LOURENÇO, Luciano & MATEUS, Manuel (coord.): Riscos Naturais, Antrópicos e Mistos, Coimbra, Departamento de Geografia da Universidade de Coimbra, 821-828.

43. FICHMAN, Robert G. (1992) – Information Technology Diffusion: A Review of Empirical Research. In *Proceedings of the Thirteenth International Conference on Information Systemns*, Dallas, Junho 1992 (195-206).

44. FITZGERALD, Brian (2006) – "The Transformation of Open Source *Software*". **MIS Quarterly**, Vol. 30, N. º 3, 587-598.

45. FITZGERALD, Brian (2011) – "Open Source *Software* Adoption: Anatomy of Success and Failure", *in* KOCH, Stefan (eds.): Multi-Disciplinary Advancement in Open Source *Software* and Processes, Nova Iorque, Information Science Reference, 1-23.

46. FITZEGERALD, Brian; KESAN, Jay; RUSSO, Barbara; SHAIKH, Maha & SUCCI, Giancarlo (2011) – **Adopting Open Source *Software* - A Practical Guide**. Massachusetts Institute of Technology, 1ª Edição, 164 p.

47. FRAMBACH, Ruud & SCHILLEWAERT, Niels (2002) – "Organizational innovation adoption: A multi-level framework of determinants and opportunities for future research". **Journal of Business Research**, N. º 55, 163-176.

48. FUGGETTA, Alfonso (2003) – "Open source *software* – an evaluation". **The Journal of Systems and *Software***, N. º 66, 77-90.

49. GALLEGO, M. Dolores; BUENO, Salvador; RACERO, F. José & NOYES, Jan (2015) – "Open source *software*: The effects of training on acceptance". **Computers in Human Behavior**, N. º 49, 390-399.

50. GALPIN, Timothy (2000) – **O lado humano da mudança: um guia prático para a mudança organizacional**. Sílabo, 1ª Edição, Lisboa, 194 p.

51. GARCIA, Ricardo (2012) – *Metodologias de Avaliação da Perigosidade e Risco associado a Movimentos de Vertente – Aplicação na bacia do rio Alenquer*. **Dissertação de Doutoramento**, Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa, 430 p.

52. GAY, Joshua (2002) – **Free *software*, free society: Selected essays of Richard Stallman**. GNU Press, Boston, 223 p.

53. GERRARD, Paul & JOHNSON, Radia (2015) – **Mastering Scientific Computing with R**. Packt Publishing, 1ª Edição, Birmingham, 414 p.

54. GHAPANCHI, Amir Hossein & AURUM, Aybuke (2012) – "The impact of project capabilities on project performance: Case of open source *software* projects". **International Journal of Project Management**, N. º 30, 407-417.

55. GLYNN, Eugene; FITZGERALD, Brian & EXTON, Chris (2005) – Commercial Adoption of Open Source *Software*: An Empirical Study. In *Proceedings of International Conference on Empirical Software Engineering*, Noosa Heads, 17-18 Novembro 2005 (225-234).

56. GODINHO, Rui (2012) – *O Open Source Software na Administração Pública Central Portuguesa – Grau e tendências de utilização; Factores inibidores e facilitadores da sua adopção*. **Dissertação de Mestrado**, ISCTE – Instituto Universitário de Lisboa, 212 p.

57. GOODCHILD, Michael (2009a) – "Geographic information systems and science: today and tomorrow". **Procedia Earth and Planetary Science**, N. º 1, 1037-1043.

58. GOODCHILD, Michael (2009b) – "NeoGeography and the nature of geographic expertise". **Journal of Location Based Services**, Vol. 3, N. º 2, 82-96.

59. GOODCHILD, Michael (2010) – "Twenty years of progress: GIScience in 2010". **Journal of Spatial Information Science**, N. º 1, 3-20.

60. GRAHAM, Mark (2010) – "Neogeography and the palimpsests of place: WEB 2.0 and the construction of a virtual earth". **Tijdschrift voor Economische en Sociale Geografie**, Vol. 101, N. º 4, 422-436.

61. GRANELL, Carlos; FERNÁNDEZ, Óscar & DÍAZ, Laura (2014) – "Geospatial information infrastructures to address spatial needs in health: Collaboration, challenges and opportunities". **Future Generation Computer Systems**, N.º 31, 213-222.

62. GRATIER, Thomas; SPENCER, Paul & HAZZARD, Erik (2015) – **OpenLayers 3 Beginner's Guide**. Packt Publishing, 2ª Edição, Birmingham, 488 p.

63. HAMLET, Dick (2015) – "Theory of *Software* Testing with Persistent State". **IEE Transactions on Reliability**, Vol. 63, N.º 3, 1098-1115.

64. HAUGE, Øyvind; AYALA, Claudia & CONRADI, Reidar (2010) – "Adoption of open source *software* in *software*-intensive organizations – A systematic literature review". **Information and *Software* Technology**, N. º 52, 1133-1154.

65. HENLEY, Mark & KEMP, Richard (2008) – "Open Source *Software*: An introduction". **Computer Law & Security Report**, N. º 24, 77-85.

66. HUTCHINSON, J. N. (1988) – "General report: Morphological and geothecnical parameters of landslide in relation to geology and hydrogeology"SCHU, *in* BONNARD, C. (Eds.): Landslides, Proceddings of the Fifth International Symposium on Landslides, Roterdão, Balkema, 3-35.

67. HWANG, Soo-yeon (2005) – "Adopting Open Source and Open Standards in the Public Sector: Five Deciding Factors behind the Movement". **Michigan Journal of Public Affairs**, Vol. 2. **Disponível online** no endereço URL: http://mjpa.umich.edu/files/2014/06/2005-Hwang-OpenSource.pdf (acedido em 29 Março, 2015).

68. JOSEPH, Alun & PHILLIPS, David (1984) – **Accessibility & Utilization – Geographical perspectives on health care delivery**. Harper & Row, 1ª Edição, Londres, 214 p.

69. JULIÃO, Rui Pedro (2001) – *Tecnologias de Informação Geográfica e Ciência Regional – Contributos Metodológicos para a Definição de Modelos de Apoio à Decisão em Desenvolvimento Regional*. **Dissertação de Doutoramento**, Faculdade de Ciências Humanas da Universidade Nova de Lisboa, 328 p.

70. JULIÃO, Rui Pedro; NERY, Fernanda; RIBEIRO, José Luís; BRANCO, Margarida & ZÊZERE, José Luís (2009) – **Guia metodológico para a produção de cartografia municipal de risco e para a criação de Sistemas de Informação Geográfica (SIG) de base municipal**. Associação Nacional de Protecção Civil, Lisboa, 91 p.

71. KANEWALA, Upulee & BIEMAN, James (2014) – "Testing scientific *software*: A systematic literature review". **Information and *Software* Technology**, N.º 56, 1219-1232.

72. KANUGANTI, Shalini; SARKAR, Ashoke; SINGH, Ajit & ARKATKAR, Shriniwas (2014) – "Quantification of accessibility to health facilities in rural areas". **Case Studies on Transport Policy,** http://dx.doi.org/10.1016/j.cstp.2014.08.004. ISSN 2213-624X.

73. KEMP, Richard (2009) – "Current developments in Open Source *Software*". **Computer Law & Security Review**, N.º 25, 569-582.

74. KEMPKA, Joseph; MCMINN, Phil & SUDHOLT, Dirk (2015) – "Design and analysis of different alternating variable searches for search-based *software* testing". **Theoretical Computer Science**, http://dx.doi.org/10.1016/j.tcs.2014.12.009.

75. KENNEDY, Michael (2009) – **Introducing Geographic Information Systems with ArcGIS: A Workbook Approach to Learning GIS**. John Wiley & Sons, 2ª Edição, 571 p.

76. KONECNY, Gottfried (2003) – **Geoinformation: Remote Sensing, Photogrammetry and Geographic Information Systems**. Taylor & Francis, 1ª Edição, Londres, 248 p.

77. KOVÁCS, George; DROZDIK, Sylvester; ZULIANI, Paolo & SUCCI, Giancarlo (2004) – Open Source *Software* for the Public Administration. In *Proceedings of the 6th International Workshop on Computer Science and Information Technologies*, Budapeste, **Disponível online** no endereço URL: http://cospa.case.unibz.it/Assets/documents/Articles/Open%20Source%20*Software*%20for%20the%20Public%20Administration.pdf (acedido em 28 Março, 2015).

78. KROGH, Georg von & HIPPEL, Eric von (2006) – "The Promise of Research on Open Source *Software*". **Management Science**, Vol. 52, N.º 7, 975-983.

79. KROGH, Georg von & SPAETH, Sebastian (2007) – "The open source *software* phenomenon: Characteristics that promote research". **Journal of Strategic Information Systems**, N. º 16, 236-253.

80. KULVIWAT, Songpol; BRUNER, Gordon & AL-SHURIDAH, Obaid (2009) – "The role of social influence on adoption of high tech innovations: The moderating effect of public/private consumption". **Journal of Business Research**, N.º 62, 706-712.

81. KUWATA, Yoshitaka; TAKEDA, Kentaro & MIURA, Hiroshi (2014) – "A study on maturity model of open source *software* community to estimate the quality of products". **Procedia Computer Science**, N. º 35, 1711-1717.

82. KWAN, Stephen & WEST, Joel (2005) – "A Conceptual Model for Enterprise Adoption of Open Source *Software*", *in* BOLIN, Sherrie (eds.): The Standards Edge: Open Season, The Bolin Group, 51-62.

83. LAURENT, Andrew (2004) – **Understanding Open Source & Free *Software* Licensing**. O'Reilly Media, 1ª Edição, Sebastopol, 207 p.

84. LAWHEAD, Joel (2015) – **QGIS Python Programming Cookbook**. Packt Publishing, 1ª Edição, Birmingham, 315 p.

85. LEI, Xiaohui; WANG, Yuhui; LIAO, Weihong; JIANG Yunzhong; TIAN Yu & WANG Hao (2011) – "Development of efficient and cost-effective distributed hydrological modelling tool MWEasyDHM based on open-source MapWindow GIS". **Computers & Geosciences**, N. º 37, 1476-1489.

86. LERNER, Josh & TIROLE, Jean (2001) – "The open source movement: Key research questions". **European Economic Review**, N. º 45, 819-826.

87. LI, Yan; TAN, Chuan-Hoo & TEO, Hock-Hai (2012) – "Leadership characteristics and developers motivation in open source *software* development". **Information & Management**, N. º 49, 257-267.

88. LINDBERG, Van (2008) – **Intellectual Property and Open Source – A Practical Guide to Protecting Code**. O'Reilly Media, 1ª Edição, Sebastopol, 371 p.

89. LONGLEY, Paul; GOODCHILD, Michael; MAGUIRE, David & RHIND, David (2005) – **Geographical Information Systems and Science**. John Wiley & Sons, 2ª Edição, Chichester, 537 p.

90. LUKASZEWSKI, Albert (2010) – **MySQL for Python**. Packt Publishing, 1ª Edição, Birmingham, 417 p.

91. LUO, Wei & WANG, Fahui (2003) – "Measures of spatial accessibility to health care in a GIS environment: synthesis and a case study in the Chicago Region". **Environment and Planning B: Planning and Design**, Vol. 30, 865-884.

92. MAGUIRE, David (1991) – "An Overview and definition of GIS", *in* MAGUIRE, David; GOODCHILD, Michael & RHIND, David (eds.): Geographical Information Systems – Principles and Applications (Vol. 1), Harlow, Longman Group, 9-20.

93. MAHONY, G. & NAUGHTON, E. (2004) – "Open Source *Software* Monetized: Out of the Bazaar and into Big Business". **The Computer & Internet Lawyer**, Vol. 21, N.º 10, 1-17.

94. MALCZEWSKI, Jacek (1999) – **GIS and multicriteria decision analysis**. John Wiley & Sons, 1ª Edição, Nova Iorque, 408 p.

95. MARANHÃO, Vanessa (2013) – *Modelagem e controle de qualidade de uma infraestrutura de dados espaciais para o Estado de Pernambuco*. **Dissertação de mestrado**, Centro de Tecnologia e Geociências da Universidade Federal de Pernambuco, 140 p.

96. MARSAN, Josianne; PARÉ, Guy & WYBO, Michael (2012a) – "Has open source *software* been institutionalized in organizations or not?" **Information and *Software* Technology**, N. º 54, 1308-1316.

97. MARSAN, Josianne; PARÉ, Guy & BEAUDRY, Anne (2012b) – "Adoption of open source *software* in organizations: A socio-cognitive perspective". **Journal of Strategic Information Systems**, N. º 21, 257-273.

98. MARSAN, Josianne & PARÉ, Guy (2013) – "Antecedents of open source *software* adoption in health care organizations: A qualitative survey of experts in Canada". **International Journal of Medical Informatics**, N. º 82, 731-741.

99. MARTIN, D.; WRIGLEY, H.; BARNETT, S. & RODERICK, P. (2002) – "Increasing the sophistication of access measurement in a rural healthcare study". **Health & Place**, N.º 8, 3-13.

100. MATTHEW, Neil & STONES, Richard (2006) – **Beginning Databases with PostgreSQL From Novice to Professional**. Apress, 2ª Edição, Nova Iorque, 637 p.

101. MATOS, João (2008) – **Fundamentos de Informação Geográfica.** LIDEL – Edições Técnicas, 1ª Edição, Lisboa, 405 p.

102. MCGRAIL, Mathew & HUMPHREYS, John (2009) – "Measuring spatial accessibility to primary care in rural areas: Improving the effectiveness of the two-step floating catchment area method". **Applied Geography**, N. º 29, 533-541.

103. MCGRAIL, Mathew & HUMPHREYS, John (2014) – "Measuring spatial accessibility to primary health care services: Utilising dynamic catchment sizes". **Applied Geography**, N. º 54, 182-188.

104. MICHAEL, Christopher & AMES, Daniel (2007) – "Evaluation of the OGC Web Processing Service for Use in a Client-Side GIS". **The Journal of the Open Source Geospatial Foundation**, Vol. 1, disponível no endereço URL: https://journal.osgeo.org/index.php/journal/issue/view/17 (acedido em 25 Agosto, 2015).

105. MITRE (2003) – Use of Free and Open Source *Software* (FOSS) in the U.S. Department of Defense. 168 p. The MITRE Corporation & The Defense Information Systems Agency. **Disponível no** endereço URL: http://dodcio.defense.gov/Portals/0/Documents/FOSS/dodfoss_pdf.pdf (acedido em 29 Agosto, 2015).

106. MOMJIAN, Bruce (2001) – **PostgreSQL: Introduction and Concepts**. Addison-Wesley, 2ª Edição, Upper Saddle River, 453 p.

107. MYERS, Glenford; BADGETT, Tom & SANDLE, Corey (2012) - **The art of *Software* Testing.** John Wiley & Sons, 3ª Edição, Nova Jersey, 240 p.

108. NETELER, Markus & MITASOVA, Helena (2008) – **Open Source GIS: A GRASS GIS Approach**. Springer, 3ª Edição, Nova Iorque, 406 p.

109. NGUI, André & VANASSE, Alain (2012) – "Assessing spatial accessibility to mental health facilities in an urban environment". **Spatial and Spatio-temporal Epidemiology**, N.º 3, 195-203.

110. NIXON, Robin (2009) – **Learning PHP, MySQL, and JavaScript**. O'Reilly, 1ª Edição, Sebastopol, 505 p.

111. OBE, Regina & HSU, Leo (2011) – **PostGIS in Action**. Manning Publications, 1ª Edição, Stanford, 492 p.

112. OBE, Regina & HSU, Leo (2012) – **PostgreSQL: Up and Running**. O'Reilly Media, 1ª Edição, Sebastopol, 145 p.

113. OLSON, Michael (2006) – "Dual Licensing", *in* DIBONA, Chris; COOPER, Danese & STONE, Mark (eds.): Open Source 2.0 – The Continuing Evolution, Sebastopol, O'Reilly Media, 71-90.

114. OPENSHAW, Stan (1991) – "A spatial analysis research agenda", *in* MASSER, Ian & BLAKEMORE, Michael (eds.): Handling Geographical Information – Methodology and Potential Applications, Harlow, Longman Group, 18-37.

115. ORAM, Andrew (2011) – "Promoting Open Source *Software* in Government: The Challenges of Motivation and Follow-Through". **Journal of Information Technology & Politics**, N. º 8, 240-252.

116. OSEPA (2012a) – Good practices in FOSS usage by European public administrations: practical guide and tools. 39 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

117. OSEPA (2012b) – OSEPA CP3 Expert Survey Conclusions. 58 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

118. OSEPA (2012c) – Good Practice Guide covering various aspects of FOSS usage by European Public Administrations. 81 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

119. OSEPA (2012d) – Background analysis report on potential indicators and benchmarks of FOSS usage assessment by European Public Administrations. 21 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

120. OSEPA (2012e) – Synthesis report on the OSEPA survey results, based on qualitative factors. 38 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

121. OSEPA (2012f) – FOSS European and National Policies and practices: Analysis and Recommendations. 58 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

122. OSEPA (2012g) – Policy Recommendation Paper, 3º version. 114 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

123. OSEPA (2012h) – Technical efficiency guidelines for selecting between and among FOSS and proprietary SW solutions. 82 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

124. OSEPA (2012i) – Guidelines for the procurement of F/OSS. 45 p. Open Source *software* usage by European Public Administrations. Atenas. Grécia. **Disponível no** endereço URL: http://osepa.eu/pdeliverables/index.php (acedido em 10 Junho, 2015).

125. ØSTERLIE, Thomas & JACCHERI, Letizia (2007) – A Critical Review of *Software* Engineering Research on Open Source *Software* Development. In *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design*, Gdansk, 5 Junho 2007 (1-10).

126. PAPADOPOULOS, Thanos; STAMATI, Teta; NIKOLAIDOU, Mara & ANAGNOSTOPOULOS, Dimosthemis (2013) – "From Open Source to Open Innovation practices: A case in the Greek context in light of the debt crisis". **Technological Forecasting & Social Change**, N. º 80, 1232-1246.

127. PATRIARCA, Joaquim; CANILHO, Sara; SACRAMENTO, João André; CORREIA, Ricardo; CASTRO, António; SANTOS, Sara; SANTOS, José Gomes & PINHO, Ricardo (2014a) – Jangada de SIG na Administração Pública Portuguesa. In *Actas das I Jornadas Lusófonas de Ciências e Tecnologias de Informação Geográfica*, Coimbra, 11 - 13 Setembro 2014 (551-581).

128. PATRIARCA, Joaquim; SANTOS, José Gomes & CANILHO, Sara (2014b) – Modelos de regressão múltipla *vs* Modelos semi-quantitativos na produção de cartografia de perigosidade geomorfológica: estudo de caso na região de Peso da Régua, Bacia do Douro – Norte de Portugal. In *Actas do XIV Colóquio Ibérico de Geografia: 'A Jangada de Pedra' – Geografias ibero-afro-americanas,* Guimarães, 11 – 14 Novembro 2014 (1633-1639).

129. PEREIRA, Alexandre (2012) – "Normas Abertas nos Sistemas Informáticos do Estado: Quo Vadis?". **Revista do Centro de Estudos de Direito do Ordenamento, do Urbanismo e do Ambiente**, N.º 1/15, 39-43.

130. PEREIRA, S.; ZÊZERE, José Luís & BATEIRA, Carlos (2012) – "Technical Note: Assessing predictive capacity and conditional Independence of landslide predisposing factores for shallow landslide susceptibility models". **Natural Harzards and Earth System Sciences**, N.º 12, 979-988.

131. PERRY, Mark & MARGONI, Thomas (2010) – FLOSS for the Canadian Public Sector: open democracy. In *Fourth International Conference on Digital Society*, St. Maarten, 10-16 Novembro 2010 (294-300).

132. PHILLIPS, Douglas E. (2009) – **Unveiled – How Legislation by License Controls *Software* Access**. Oxford University Press, 1ª Edição, Nova Iorque, 204 p.

133. PHILIPS, Steven; ANDERSON, Robert & SCHAPIRE, Robert (2006) – "Maximum entropy modelling of species geographic distributions". **Ecological Modelling**, N. º 190, 231-259.

134. PIEDADE, Aldina; ZÊZERE, José Luís; TENEDÓRIO, José; GARCIA, Ricardo; OLIVEIRA, Sérgio; ROCHA, Jorge (2011) – "Modelação em Sistemas de Informação Geográfica da avaliação da susceptibilidade a movimentos de vertente na área amostra de Lousa-Loures (Região Norte de Lisboa)", *in* SANTOS, Noberto & CUNHA, Lúcia (coord.): Trunfos de uma Geografia Activa – desenvolvimento local, ambiente, ordenamento e tecnologia, Coimbra, Imprensa da Universidade de Coimbra, 539-546.

135. QU, Wen Guang; YANG, Zhiyong & WANG, Zhongming (2011) – "Multi-level framework of open source *software* adoption". **Journal of Business Research**, N. º 64, 997-1003.

136. QUICK, John (2010) – **Statistical Analysis with R: Beginner's Guide**. Packt Publishing, 1ª Edição, Birgmingham, 282 p.

137. RADTKE, Nicholas; JANSSEN, Marco & COLLOFELLO, James (2011) – "What Makes Free/Libre Open Source *Software* (FLOSS) Projects Successful? An Agent-Based Model of FLOSS Projects", *in* KOCH, Stefan (eds.): Multi-Disciplinary Advancement in Open Source *Software* and Processes, Nova Iorque, Information Science Reference, 87-98.

138. RAFOSS, Trond; SÆLID, Knut; GYLAND, Lars & ENGRAVSLIA, Liv (2010) – "Open geospatial technology standards and their potential in plant pest risk management (GPS enabled mobile phones utilising open geospatial technology standards Web Feature Transactions support the fighting of fire blight in Norway". **Computers and Electronics in Agriculture**, N.º 74, 336-340.

139. RAUTENBACH, Victoria; COETZEE, Serena & IWANIAK, Adam (2013) – "Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure". **Computers, Environment and Urban Systems**, N.º 37, 107-120.

140. RAYMOND, Eric (2001) – **The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition**. O'Reilly, 2ª Edição, Sebastopol, 233 p.

141. REISINGER, Markus; RESSNER, Ludwing; SCHMIDTKE, Richard & THOMES, Tim Paul (2014) – "Crowding-in of complementary contributions to public goods: Firm investment into open source *software*". **Journal of Economic Behavior & Organization**, N. º 106, 78-94.

142. RIEHLE, Dirk (2010) – "The Economic Case for Open Source Foundations". **Disponível online** no endereço URL: http://dirkriehle.com/publications/2010-2/the-economic-case-for-open-source-foundations/ (acedido em 9 Abril, 2015).

143. RIGGS, Simon & KROSING, Hannu (2010) – **PostgreSQL 9 Administration Cookbook**. Packt Publishing, 1ª Edição, Birmingham, 345 p.

144. ROCHA, Fernando (2012) – *Sistemas Complexos, Modelação e Geosimulação da Evolução de Padrões de Uso e Ocupação do Solo*. **Dissertação de Doutoramento**, Instituto de Geografia e Ordenamento do Território da Universidade de Lisboa, 954 p.

145. RODRIGUES, Albano (2013) – *Assessing impacts from future climatic scenarios on the distribution of flora and vegetation at Madeira Island*. **Dissertação de Doutoramento**, Faculdade de Letras da Universidade de Coimbra, 331 p.

146. RODRIGUES, Ricardo (1998) – **Organizações, Mudança e Capacidade de Gestão.** Principia, Cascais, 128 p.

147. ROGERS, Everett M. (1983) – **Diffusion of Innovation**. The Free Press, 3ª Edição, Nova Iorque, 453 p.

148. ROSERO-BIXBY, Luis (2004) – "Spatial access to health care in Costa Rica and its equity: a GIS-based study". **Social Science & Medicine**, N.º 58, 1271-1284.

149. SAATY, Thomas (1988) – **The analytic hierarchy process: planning, priority setting, resource allocation**. Mcgraw-Hill, ISBN 0070543712.

150. SANDERSON, Mike; STICKLER, Graham & RAMAGE, Steven (2007) – "1Spatial: Spatial Data Quality and the Open Source Community". **The Journal of the Open Source Geospatial Foundation**, Vol. 2, disponível no endereço URL: https://journal.osgeo.org/index.php/journal/issue/view/21 (acedido em 25 Agosto, 2015).

151. SANTOS, José Gomes (2002) – "Cartografia automática do Risco de Movimentos de Vertente; Estudo aplicado à área de Peso da Régua Bacia do Douro – Norte de Portugal". **Revista de Xeografia, Territorio e Medio Ambiente**, N. º 2, 33-57.

152. SANTOS, José Gomes (2013) – "GIS-based hazard and risk maps of the Douro river basin (north-eastern Portugal)". **Geomatics, Natural Hazards and Risk,** 1-25. doi:10.1080/197475705.2013.831952

153. SARRAB, Mohamed & REHMAN, Osama (2014) – "Empirical study of open source *software* selection for adoption, based on *software* quality characteristics". **Advances in Engineering *Software***, N. º 69, 1-11.

154. SÁTIRO, Talita & SIMÕES, Sílvio (2013) – Comparação entre dois Sistemas de Informação Geográfica (ArcGIS e gvSIG) na elaboração de um mapa de potencialidade para a silvicultura baseado em elementos do meio físico – a bacia do rio Paraíba do Sul (Porção Paulista). In *Anais XVI Simpósio Brasileiro de Sensoriamento Remoto*, Foz do Iguaçu, 13-18 de Abril 2013 (5147-5154).

155. SCACCHI, Walt; FELLER, Joseph; FITZGERALD, Brian; HISSAM, Scott & LAKHANI, Karim (2006) – "Understanding Free/Open Source *Software* Development Processes". ***Software* Process Improvement and Practice**, N. º 11, 95-105.

156. SELTZER, Wendy (2006) – "Why Open Source Needs Copyright Politics", *in* DIBONA, Chris, COOPER, Danese & STONE, Mark (eds.): Open Source 2.0 – The Continuing Evolution, Sebastopol, O'Reilly Media, 149-159.

157. SHERMAN, Gary (2008) – **Desktop GIS Mapping the Planet with Open Source Tools.** Pragmatic Bookshelf, 1ª Edição, 349 p.

158. SINGH, Vandana & HOLT, Lila (2013) – "Learning and best practices for learning in open-source *software* communities". **Computers & Education**, N. º 63, 98-108.

159. SMITH, Gregory (2010) – **PostgreSQL 9.0 High Performance**. Packt Publishing, 1ª Edição, Birmingham, 442 p.

160. SMITH, Kevin (2004) – **Environment Hazards – Assessing Risk and Reducing Disaster**. Routledge, 4ª Edição, Londres, 306 p.

161. SOUZA, Bruno (2006) – "How Much Freedom Do You Want?", *in* DIBONA, Chris; COOPER, Danese & STONE, Mark (eds.): Open Source 2.0 – The Continuing Evolution, Sebastopol, O'Reilly Media, 211-228.

162. SPINELLIS, Diomidis & GIANNIKAS, Vaggelis (2012) – "Organizational adoption of open source *software*". **The Journal of Systems and *Software***, N. º 85, 666-682.

163. STEINIGER, Stefan & BOCHER, Erwan (2009) – "An overview on current Free and Open Source Desktop GIS developments". **International Journal of Geographical Information Science**, Vol. 23, Issue 10, 1345-1370.

164. STEINIGER, Stefan & HAY, Geoffrey J. (2009) – "Free and open source geographic information tools for landscape ecology". **Ecological Informatics**, N. º 4, 183-195.

165. STEINIGER, Stefan & HUNTER, Andrew (2013) – "The 2012 free and open source GIS *software* map – A guide to facilitate research, development, and adoption". **Computers, Environment and Urban Systems**, N. º 39, 136-150.

166. STOL, Klaas-Jan & BABAR, Muhammad (2009) – "Reporting Empirical Research in Open Source *Software*: The State of Practice". In *Open Source Ecosystems: Diverse Communities Interacting – 5th IFIP WG 2.13 International Conference on Open Source Systems*, Skövde, 3-6 Junho 2009 (156-169).

167. SUBRAMANYAM, Ramanath & XIA, Mu (2008) – "Free/Libre Open Source *Software* development in developing and developed countries: A conceptual framework with an exploratory study". **Decision Support Systems**, N. º 46, 173-186.

168. TALUKDER, Majharul (2012) – "Factors affecting the adoption of technological innovation by individual employees: An Australian study". **Social and Behavioral Sciences**, N.º 40, 52-57.

169. TANSER, Frank; GIJSBERTSEN, Brice & HERBST, Kobus (2006) – "Modelling and understanding primary health care accessibility and utilization in rural South Africa: An exploration using a geographical information system". **Social Science & Medicine**, N. º 63, 691-705.

170. TORNATZKY, Louis; FLEISCHER, Mitchell & CHAKRABARTI, Alok (1990) – **The processes of technological innovation.** Lexington Books, Massachusetts, 298 p.

171. UNTERKALMSTEINER, Michael; GORSCHEK, Tony; FELDT, Robert & KLOTINS, Eriks (2015) – "Assessing requirements engineering and *software* test alignment – Five case studies". **The Journal of Systems and *Software***, N.º 109, 62-77.

172. VANDERHAEGEN, Marc & MURO, Eva (2005) – "Contribution of a European spatial data infrastructure to the effectiveness of EIA and SEA studies". **Environmental Impact Assessment Review**, N.º 25, 123-142.

173. VARNES, D. J. (1978) – "Slope Movement Types and Processes", *in* SCHUSTER, R. & KRIZEK, R. (Eds.): Landslides, Analysis and Control, Washington D. C., Transportation Research Board Special Report 176, 11-33.

174. VARNES, D. J. (1984) – **Landslide hazard zonation: a review of principles and practice**. UNESCO, Paris, 63 p.

175. VEN, Kris & VERELST, Jan (2010) – "The Impact of Ideology on the Organizational Adoption of Open Source *Software*", *in* SIAU, Keng & ERICKSON, John (coord.): Principle Advancements in Database Management Technologies: New Applications and Frameworks, Nova Iorque, Information Science Reference, 160-175.

176. VEN, Kris & VERELST, Jan (2011) – "An Empirical Investigation into the Assimilation of Open Source Server *Software*". **Communications of the Association for Information Systems**, Vol. 28, N.º 9, 118-137.

177. WANG, Jing (2012) – "Survival factors for Free Open Source *Software* projects: A multi-stage perspective". **European Management Journal**, N. º 30, 352-371.

178. WAYNER, Peter (2000) – **Free For All – How Linux and the Free *Software* Movement Undercut the High Tech Titans**. Haper Business, Nova Iorque, 340 p.

179. WEIBEL, Robert & HELLER, M. (1991) – "Digital Terrain Modelling", *in* MAGUIRE, David; GOODCHILD, Michael & RHIND, David (eds.): Geographical Information Systems – Principles and Applications (Vol. 1), Harlow, Longman Group, 269-297.

180. WESTRA, Erik (2014) – **Building Mapping Applications with QGIS**. Packt Publishing, 1ª Edição, Birmingham, 248 p.

181. WILLIAMS, Sam (2010) – **Free as in Freedom (2.0): Richard Stallman and the Free *Software* Foundation**. Free *Software* Foundation, 2ª Edição, Boston, 229 p.

182. WISE, Lyndsay (2012) – **Using Open Source Platforms for Business Intelligence – Avoid Pitfalls and Maximize ROI**. Elsevier, Waltham, 210 p.

183. WISE, Steve (1998) – "The Effect of GIS Interpolation Errors on the Use of Digital Elevation Models in Geomorphology", *in* LANE, Stuart; RICHARDS, Keith & CHANDLER, Jim (eds.): Landform Monitoring, Modelling and Analysis, Chichester, John Wiley & Sons, 139-164.

184. YANK, Kevin (2009) – **Build your own database driven WEB Site using PHP & MySQL**. SitePoint Pty., 4ª Edição, Collingwood, 479 p.

185. YAO, Jing; MURRAY, Alan & AGADJANIAN, Victor (2013) – "A geographical perspective on access to sexual and reproductive health care for women in rural Africa". **Social Science & Medicine**, N. º 96, 60-68.

Legislação Consultada e documentos/directivas europeias

1. Comunicação da Comissão ao Conselho, Parlamento Europeu, Comité Económico e Social e Comité das Regiões – "eEurope 2005: Uma sociedade de informação para todos". Plano de Acção a apresentar com vista ao Conselho Europeu de Sevilha, 21-22 de Junho de 2002.
2. Decreto-Lei n.º 309/93 de 2 de Setembro. Diário da República, 1.ª Série – N.º 206 – 2 de Setembro de 1993. Lisboa. Portugal.
3. Decreto-Lei n.º 172/95 de 18 de Julho. Diário da República, 1.ª Série – N.º 164 – 18 de Julho de 1995. Lisboa. Portugal.
4. Decreto-Lei n.º 193/95 de 28 de Julho. Diário da República, 1.ª Série-A – N.º 173 – 28 de Julho de 1995. Lisboa. Portugal.
5. Decreto-Lei n.º 270/2001 de 6 de Outubro. Diário da República, 1.ª Série – N.º 232 – 6 de Outubro de 2001. Lisboa. Portugal.
6. Decreto-Lei n.º 86/2002 de 6 de Abril. Diário da República, 1.ª Série – N.º 81 – 6 de Abril de 2002. Lisboa. Portugal.
7. Decreto-Lei n.º 202/2007 de 25 de Maio. Diário da República, 1ª Série – N.º 101 – 25 de Maio de Maio de 2007.
8. Decreto-Lei n.º 129/2008 de 21 de Julho. Diário da República, 1.ª Série – N.º 139 – 21 de Julho de 2008. Lisboa. Portugal.
9. Decreto-Lei n.º 142/2008 de 24 de Julho. Diário da República, 1.ª Série – N.º 142 – 24 de Julho de 2008. Lisboa. Portugal.
10. Decreto-Lei n.º 166/2008 de 22 de Agosto. Disponível no endereço URL http://www.pgdlisboa.pt/leis/lei_mostra_articulado.php?tabela=leis&artigo_id=&nid=1058&nversao=&tabela=leis&so_miolo= (acedido em 6 Agosto, 2015). Lisboa. Portugal.
11. Decreto-Lei n.º 46/2009 de 20 de Fevereiro. Diário da República, 1.ª Série – N.º 36 – 20 de Fevereiro de 2009.
12. Decreto-Lei n.º 34/2012 de 14 de Fevereiro. Diário da República, 1.ª Série – N.º 32 – de 14 de Fevereiro de 2012. Lisboa. Portugal.
13. Decreto-Lei n.º 107/2012 de 18 de Maio. Diário da República, 1.ª Série – N.º 97 – de 18 de Maio de 2012. Lisboa. Portugal.

14. Decreto-Lei n.º 145/2012 de 11 de Julho. Diário da República, 1.ª Série – N.º 133 – de 11 de Julho de 2012. Lisboa. Portugal.

15. Decreto-Lei n.º 180/2009 de 7 de Agosto. Diário da República, 1.ª Série – N.º 152 - 7 de Agosto de 2009. Lisboa. Portugal.**0**

16. Decreto Regulamentar n.º 10/2009 de 29 de Maio. Diário da República, 1.ª Série – N.º 104 – 29 de Maio de 2009. Lisboa. Portugal.

17. Decreto Regulamentar n.º 11/2009 de 29 de Maio. Diário da República, 1.ª Série – N.º 104 – 29 de Maio de 2009. Lisboa. Portugal.

18. Decreto Regulamentar n.º 142/2012 de 26 de Janeiro. Diário da República, 1.ª Série – N.º 19 – 26 de Janeiro de 2012. Lisboa. Portugal.

19. Decreto Regulamentar n.º 32/2012 de 20 de Março. Diário da República, 1.ª Série – N.º 57 – 20 de Março de 2012. Lisboa. Portugal.

20. Deliberação n.º 1495/2013 de 24 de Julho. Diário da República, 2.ª Série – N.º 141 – 24 de Julho de 2013. Lisboa. Portugal.

21. Directiva 2007/2/CE do Parlamento Europeu e do Conselho que estabelece uma infra-estrutura de informação geográfica na Comunidade Europeia (INSPIRE), de 14 de Março de 2007. Bruxelas. Bélgica.

22. Lei n.º 3/2004 de 15 de Janeiro. Diário da República, 1.ª Série - A – N.º 12 – 15 de Janeiro de 2004. Lisboa. Portugal.

23. Lei n.º 4/2004 de 15 de Janeiro. Diário da República, 1.ª Série – A – N.º 12 – 15 de Janeiro de 2004. Lisboa. Portugal.

24. Lei n.º 27/2006 de 3 de Julho. Diário da República, 1.ª Série – N.º 126 – 3 de Julho de 2006. Lisboa. Portugal.

25. Lei n.º 65/2007 de 12 de Novembro. Diário da República, 1ª Série – N.º 217 – 12 de Novembro de 2007. Lisboa. Portugal.

26. Lei n.º 36/2011 de 21 de Junho. Diário da República, 1.ª Série – N.º 18 - 21 de Junho de 2011. Lisboa. Portugal.

27. Lei n.º 66-B/2012 de 31 de Dezembro (Orçamento do Estado para 2013). Diário da República, 1.ª Série – N.º 252 – de 31 de Dezembro de 2012. Lisboa. Portugal.

28. Lei n.º 83-C/2013 de 31 de Dezembro (Orçamento do Estado para 2014). Diário da República, 1.ª Série – N.º 253 – de 31 de Dezembro de 2013. Lisboa. Portugal.

29. Portaria n.º 245/2011, de 22 de Junho. Diário da República, 1ª Série – N.º 119 – de 22 de Junho de 2011. Lisboa. Portugal.

30. Portaria n.º 224/2012, de 27 de Julho. Diário da República, 1.ª Série – N.º 145 – de 27 de Julho de 2012. Lisboa. Portugal.

31. Portaria n.º 425/2012, de 28 de Dezembro. Diário da República, 1.ª Série – N.º 251 – de 28 de Dezembro de 2012.

32. Regulamento do Parlamento Europeu e do Conselho que estabelece o programa Copernicus e revoga o Regulamento (UE) n.º 911/2010, de 29 de Maio de 2013. Bruxelas. Bélgica.

33. Regulamento do Parlamento Europeu e do Conselho, de 25 de Outubro de 2012. Bruxelas. Bélgica.

34. Regulamento n. º 329/2013, de 28 de Agosto. Diário da República, 2.ª Série – N.º 165 – de 28 de Agosto de 2013. Lisboa. Portugal.

35. Resolução da Assembleia da República n.º 66/2004. Disponível no endereço URL http://*software*livre.gov.pt/Documentacao/legislacao/ (acedido em 13 Agosto, 2015).

36. Resolução do Conselho de Ministros n.º 21/2002. Diário da República, 1.ª Série-B – N.º 26 – de 31 de Janeiro de 2002. Lisboa. Portugal.

37. Resolução do Conselho de Ministros n.º 12/2012, de 7 de Fevereiro. Diário da República, 1.ª Série – N.º 27 – 7 de Fevereiro de 2012. Lisboa. Portugal.

38. Resolução do Conselho de Ministros n.º 78/2012, de 11 de Setembro. Diário da República, 1.ª Série – N.º 1796 – de 11 de Setembro de 2012.

39. Resolução do Conselho de Ministros n.º 91/2012, de 8 de Novembro. Diário da República, 1.ª Série – N.º 2016 – de 8 de Novembro de 2012. Lisboa. Portugal.

Cartografia e Dados estatísticos

1. *Carta Administrativa Oficial de Portugal* (2014). Direcção Geral do Território. **Disponível online** no endereço URL: http://www.dgterritorio.pt/cartografia_e_geodesia/cartografia/carta_administrativa_oficial_de_portugal__caop_/caop_em_vigor/ (acedido em 10 Abril, 2015).

2. *Carta de Risco de Incêndio Florestal* - Portugal continental (2011). Autoridade Nacional da Protecção Civil e Direcção Geral dos Recursos Florestais. **Disponível online** no endereço URL: http://www.igeo.pt/scrif/crif/CRIF2011shp.zip (acedido em 11 Janeiro, 2014).

3. *Carta de Uso e Ocupação do Solo (nível 2)* – Portugal continental (2007). Direcção Geral do Território. **Disponível online** no endereço URL: http://www.dgterritorio.pt/cartografia_e_geodesia/cartografia/cos/cos__2007/ (acedido em 7 Janeiro, 2015).

4. *Carta Geológica de Portugal* – Esc. 1: 50 000. Direcção Geral de Minas e Serviços Geológicos (folha n.º 10-C).

5. *Carta Militar de Portugal* – Esc. 1:25 000. Instituto Geográfico do Exército (folhas n.º 113, 114, 115, 125, 126, 127, 136, 137, 138, 176, 177, 178, 179, 180, 181, 182, 187, 188, 189, 190, 191, 192, 193, 198, 199, 200, 201, 202, 203, 204, 207, 208, 209, 210, 211, 212, 213, 214, 215, 218, 219, 220, 221, 222, 223, 224, 225, 226, 229, 230, 231, 232, 233, 234, 235, 236, 237, 240, 241, 242, 243, 244, 245, 246, 247, 250, 251, 252, 253).

6. Contratos públicos estabelecidos entre os órgãos da APP e fornecedores de produtos/serviços relacionados com software SIG ou S4OS. **Disponível online** no endereço URL: http://www.base.gov.pt/Base/pt/Homepage (acedido em 23 Outubro, 2015).

7. *Corine Land Cover* - Portugal Continental (2006). Direcção Geral do Território. **Disponível online** no endereço URL: http://www.dgterritorio.pt/cartografia_e_geodesia/cartografia/cartografia_tematica/corine_land_cover__clc_/ (acedido em 7 Janeiro, 2015)

8. Dados relativos à distribuição da População, Edifícios e Alojamentos observados nos Censos 2011 numa quadricula 1x1 km. **Disponível online** no endereço URL: http://geogrid.ine.pt/ (acedido em 12 Julho, 2015).

9. Informação vectorial referente às ruas e rodovias do território continental português, disponibilizada pelo *Open Street Maps*. **Disponível online** no endereço URL: http://download.geofabrik.de/europe.html (acedido em 7 Janeiro, 2015).

10. Resultados do Inquérito à Utilização das Tecnologias da Informação e Comunicação na Administração Pública Central, Regional e Câmaras Municipais e indicadores actualizados da publicação Sociedade da Informação em Portugal 2012. **Disponível online** no endereço URL: http://www.dgeec.mec.pt/np4/12.html (acedido em 4 Abril, 2015).

ANEXO A – REPOSITÓRIO DE PROGRAMAS UTILIZADOS EM PARTES ESPECÍFICAS DA DISSERTAÇÃO

"O nosso contributo para a comunidade!"

Programa 1 - Bloco de código (em Python, v. 2.7.3) preparado para calcular as frequências absolutas e relativas (simples e acumuladas) de uma tabela, cujas instâncias dizem respeito a montantes gastos com aquisição de software, segundo intervalos previamente definidos pelo operador.

```
def calculo_frequencias(xls_entrada, folha_entrada, coluna, xls_saida,
intervalos):
    import xlwt, xlrd
    excel = xlrd.open_workbook(xls_entrada)
    livro = excel.sheet_by_name(folha_entrada)
    lista_valores = []
    for contrato in range(1,livro.nrows):
       cell = livro.cell(contrato, coluna)
       cell_value = cell.value
       lista_valores.append(float(cell_value))
    frequencias = []
    logos = 0
    classes = len(intervalos) + 1
    for i in range(classes):
       freq_absoluta = 0
       if logos == 0:
          for e in range(len(lista_valores)):
             if lista_valores[e] <= intervalos[i]:
                freq_absoluta += 1
       else:
          if logos != len(intervalos):
             for e in range(len(lista_valores)):
                o = i -1
                if lista_valores[e] > intervalos[o] and lista_valores[e] <= intervalos[i]:
                   freq_absoluta +=1
          else:
             for e in range(len(lista_valores)):
                if lista_valores[e] > intervalos[-1]:
                   freq_absoluta +=1
       frequencias.append(freq_absoluta)
       logos += 1
    livro = xlwt.Workbook()
    add_livro = livro.add_sheet('frequencias')
    titulos = ['classe', 'freq_abs', 'freq_abs_acu', 'freq_rel', 'freq_rel_acu']
    for i in range(len(titulos)):
       add_livro.write(0, i, titulos[i])
    contador = 0
    e = 1
    for i in range(len(frequencias)):
       add_livro.write(e, 1, frequencias[i])
       freq_relativa = (frequencias[i] / float(len(lista_valores)))*100
       add_livro.write(e, 3, freq_relativa)
       if contador == 0:
          add_livro.write(e, 2, frequencias[i])
```

```
            add_livro.write(e, 4, freq_relativa)
        else:
            temp = 0
            for freq in range(i+1):
                temp = temp + frequencias[freq]
                add_livro.write(e, 2, temp)
                freq_relativa = (temp / float(len(lista_valores)))*100
                add_livro.write(e, 4, freq_relativa)
            e +=1
            contador += 1
    livro.save(xls_saida)
```

Programa 2 - Bloco de código (em Python, v. 2.7.3) que realiza a inquirição das relações cujas instâncias dizem respeito a contratos com aquisição de SP e SL/CA, tendo em vista a criação de uma nova relação, cujas instâncias discriminam os gastos totais que cada organismo teve durante o período considerado.

```
def cria_tabela_orgaos(xls_entrada, folha_entrada, col_org, col_euros,
xls_saida):
    import xlwt, xlrd, numpy
    excel = xlrd.open_workbook(xls_entrada)
    livro = excel.sheet_by_name(folha_entrada)
    orgaos = []
    for contrato in range(1, livro.nrows):
        cell = livro.cell(contrato, col_org)
        value = cell.value
        orgaos.append(value)
    id_orgaos = numpy.unique(orgaos)
    del orgaos
    dic = {}
    for orgao in id_orgaos:
        montante = 0
        for contrato in range(1, livro.nrows):
            cell = livro.cell(contrato, col_org)
            _orgao_ = cell.value
            if int(orgao) == int(_orgao_):
                _euros_ = livro.cell(contrato, col_euros)
                euros = float(_euros_.value)
                montante += euros
        dic.update({orgao:montante})
    excel = xlwt.Workbook()
    add = excel.add_sheet('orgaos')
    add.write(0, 0, 'orgao')
    add.write(0,1, 'euros')
    nr = 1
    for orgao in dic.keys():
        add.write(nr, 0, orgao)
        add.write(nr, 1, dic[orgao])
        nr += 1
    excel.save(xls_saida)
```

Programa 3 - Bloco de código (em Python, v. 2.7.3) que procede à inquirição das relações cujas instâncias dizem respeito a contratos com aquisição de SP e SL/CA, de modo a sistematizar os totais de despesa por empresa fornecedora/sector de actividade da APP e por ano.

```
import xlrd, xlwt, numpy
def listar_elementos(folha, coluna):
     lst = []
```

```
      for contrato in range(1, folha.nrows):
         cell = folha.cell(contrato, coluna)
         value = cell.value
         lst.append(str(value))
      unicos = numpy.unique(lst)
      return unicos
def gastos_por_ano(folha, orgao, ano, coluna_orgao, coluna_ano,
coluna_montante):
      lst = []
      for contrato in range(1, folha.nrows):
         cell = folha.cell(contrato, coluna_orgao)
         if str(cell.value) == str(orgao):
            val_ano = folha.cell(contrato, coluna_ano)
            if str(val_ano.value) == str(ano):
               euros = folha.cell(contrato, coluna_montante)
               lst.append(float(euros.value))
      return sum(lst)
def __main__(xls, folha, cln_ano, cln_orgao, cln_montante, relatorio):
     excel = xlrd.open_workbook(xls)
     livro = excel.sheet_by_name(folha)
     anos = listar_elementos(livro, cln_ano)
     vertical = listar_elementos(livro, cln_orgao)
     dic_geral = {}
     for i in vertical:
        dic_vertical = {}
        for ano in anos:
           gastos = gastos_por_ano(livro, i, ano, cln_orgao, cln_ano, cln_montante)
           dic_vertical.update({int(float(ano)):gastos})
        dic_geral.update({i:dic_vertical})
     saida = xlwt.Workbook()
     add = saida.add_sheet('desp_ano')
     nr = 1
     for i in range(len(anos)):
        add.write(0, nr, int(float(anos[i])))
        nr += 1
     nr = 1
     for chave in dic_geral.keys():
        add.write(nr, 0, str(chave))
        nr += 1
     linha = 1
     for vertical in dic_geral.keys():
        coluna = 1
        for ano in anos:
           for chave in dic_geral[vertical].keys():
              if int(chave) == int(float(ano)):
                 add.write(linha, coluna, dic_geral[vertical][chave])
           coluna += 1
        linha += 1
     saida.save(relatorio)
```

Programa 4 - *Script* (em Python, v. 2.73) que executa, no ArcGIS 10.2, a associação de tabelas que nos permite gerar uma nova relação onde cada instância apresenta a localização geográfica de cada município e o total de despesas por município da sede da entidade adjudicante.

```
import arcpy, os
xls = arcpy.GetParameterAsText(0)
livro = arcpy.GetParameterAsText(1)
col_concelho = arcpy.GetParameterAsText(2)
col_montante = arcpy.GetParameterAsText(3)
caop = arcpy.GetParameterAsText(4)
def join_concelhos_contratos(tabela, folha, concelho, montante, concelhos):
```

```
    os.mkdir("C:\\areatrab")
    arcpy.ExcelToTable_conversion(tabela, "C:\\areatrab\\tabela.dbf", folha)
    arcpy.Statistics_analysis("C:\\areatrab\\tabela.dbf",
    "C:\\areatrab\\sum_concelho.dbf", [[montante, "SUM"]], concelho)
    arcpy.JoinField_management(concelhos, "FID", "C:\\areatrab\\sum_concelho.dbf",
    concelho, "")
join_concelhos_contratos(xls, livro, col_concelho, col_montante, caop)
```

Programa 5 - Bloco de código (em Python, v. 2.7.3) que viabiliza o cálculo do valor amostral médio (O) para comparação directa entre os montantes que respeitam ao SP e ao SL/CA.

```
# Selecciona iterativamente (iterações = n.º de amostras pretendidas) uma amostra
aleatória de contratos relativos ao software proprietário, em que o número de elementos
da amostra é igual ao número de contratos relativos a SLCA registados. Assim, obtém-se
uma amostra que permite a comparação directa entre os custos com proprietário e SLCA
def custo_SLCA(ficheiro, folha, nr_contratos_LCA, coluna, cenarios):
    import xlrd, random
    excel = xlrd.open_workbook(ficheiro)
    livro = excel.sheet_by_name(folha)
    lista_cenarios = []
    celulas =[]
    for contrato in range(1,livro.nrows):
       cell = livro.cell(contrato, coluna)
       cell_value = cell.value
       celulas.append(float(cell_value))
    for i in range(cenarios):
       valor_contratos = 0
       amostra = random.sample(celulas, nr_contratos_LCA)
       for i in range(len(amostra)):
          valor_contratos = valor_contratos + amostra[i]
       lista_cenarios.append(valor_contratos)
    soma = 0
    for i in lista_cenarios:
       soma = soma + float(i)
    media = soma / len(lista_cenarios)
    logos = 0
    for i in lista_cenarios:
       instancia = (i - media)**2
       if logos == 0:
          somatorio = instancia
       else:
          somatorio = somatorio + instancia
       logos += 1
    variancia = somatorio / (len(lista_cenarios) - 1)
    desvio_padrao = variancia**0.5
    print u"Valor amostral médio em euros: " + str(media)
    print u"Desvio padrão em euros: " + str(desvio_padrao)
    print u"Mínimo: " + str(min(lista_cenarios))
    print u"Máximo: " + str(max(lista_cenarios))
```

Programa 6 – Bloco de código (em Python, v. 2.7.3) que identifica, usando os recursos da biblioteca GDAL/OGR, o tipo de despesa por município.

```
from osgeo import ogr
def __main__(caop, campo_sig, campo_s4os, campo_slca):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(caop, 1)
    layer = dataSource.GetLayer()
    campo = ogr.FieldDefn("tip_desp", ogr.OFTInteger)
```

```
    layer.CreateField(campo)
    for feature in layer:
       sig = float(feature.GetField(campo_sig))
       s4os = float(feature.GetField(campo_s4os))
       slca = float(feature.GetField(campo_slca))
       if sig == 0.0 and s4os == 0.0 and slca == 0.0:
          feature.SetField("tip_desp", 0) # Sem despesas
       elif sig > 0.0 and s4os == 0.0:
          feature.SetField("tip_desp", 1) # despesas apenas com sig proprietário
       elif sig == 0.0 and s4os > 0.0:
          feature.SetField("tip_desp", 2) # despesas apenas com s4os proprietário
       elif sig == 0.0 and s4os == 0.0 and slca > 0.0:
          feature.SetField("tip_desp", 3) # despesas apenas com SLCA
       elif sig > 0.0 and s4os > 0.0:
          feature.SetField("tip_desp", 4) # despesas com sig e s4os proprietários
       layer.SetFeature(feature)
```

Programa 7 - Método (em Python, v .2.7.3) que, recorrendo à biblioteca GDAL/OGR 1.11.1, quantifica o número de órgãos que, em cada concelho que regista contratos relacionados com a aquisição/implementação de SP e SL/CA, estabeleceram, no período de tempo em análise, contratos para a aquisição/implementação de SP e SL/CA.

```
from osgeo import ogr
import xlrd
def org_s4oseOSS(shapefile, bd_s4os, folha_s4os, bd_oss, folha_oss,
id_concelho):
    driver = ogr.GetDriverByName('ESRI Shapefile')
    dataSource = driver.Open(shapefile, 1)
    layer = dataSource.GetLayer()
    linha = layer.GetNextFeature()
    campo = ogr.FieldDefn('orgao', ogr.OFTInteger)
    layer.CreateField(campo)
    bd_s4os = xlrd.open_workbook(bd_s4os)
    l_s4os = bd_s4os.sheet_by_name(folha_s4os)
    bd_oss = xlrd.open_workbook(bd_oss)
    l_oss = bd_oss.sheet_by_name(folha_oss)
    while linha:
       lista_instituicoes_s4os = []
       lista_instituicoes_oss = []
       concelho = int(str(linha.GetField(id_concelho)))
       for i in range(1, l_s4os.nrows):
          cell = l_s4os.cell(i, 6)
          municipio = int(cell.value)
          if municipio == concelho:
             cell2 = l_s4os.cell(i, 3)
             instituicao = int(cell2.value)
             lista_instituicoes_s4os.append(instituicao)
       for i in range(1, l_oss.nrows):
          cell = l_oss.cell(i, 6)
          municipio = int(cell.value)
          if municipio == concelho:
             cell2 = l_oss.cell(i,3)
             instituicao = int(cell2.value)
             lista_instituicoes_oss.append(instituicao)
       list_temp = []
       for i in range(len(lista_instituicoes_s4os)):
          if lista_instituicoes_s4os[i] not in list_temp:
             list_temp.append(lista_instituicoes_s4os[i])
       lista_instituicoes_s4os = list_temp
       list_temp = []
       for i in range(len(lista_instituicoes_oss)):
```

```
            if i == 0:
               list_temp.append(lista_instituicoes_oss[i])
            else:
               conta = 0
               for e in range(len(list_temp)):
                  if lista_instituicoes_oss[i] == list_temp[e]:
                     conta += 1
               if conta == 0:
                  list_temp.append(lista_instituicoes_oss[i])
         lista_instituicoes_oss = list_temp
         del list_temp, i
         contador = 0
         for i in range(len(lista_instituicoes_s4os)):
            for e in range(len(lista_instituicoes_oss)):
               if lista_instituicoes_s4os[i] == lista_instituicoes_oss[e]:
                  contador += 1
         linha.SetField('orgao', contador)
         layer.SetFeature(linha)
         linha = layer.GetNextFeature()
```

Programa 8 - Conjunto de métodos (em Python, v. 2.7.3) que, a partir dos contratos disponíveis, reúnem numa nova tabela as informações contratuais dos organismos que estabeleceram, no período de tempo em apreciação, contratos com fornecedores de SP e com prestadores de suporte relativo a SL/CA.

```
def id_organismo(despesa_s4os, folha_s4os, despesa_oss, folha_oss):
    import xlrd
    import numpy
    bd = xlrd.open_workbook(despesa_s4os)
    livro = bd.sheet_by_name(folha_s4os)
    lista = []
    for i in range(1, livro.nrows):
       cell = livro.cell(i, 3)
       id_org = int(cell.value)
       lista.append(id_org)
    del bd, livro, cell, id_org
    bd = xlrd.open_workbook(despesa_oss)
    livro = bd.sheet_by_name(folha_oss)
    for i in range(1, livro.nrows):
       cell = livro.cell(i, 3)
       id_org = int(cell.value)
       lista.append(id_org)
    del bd, livro, cell, id_org
    id_unico = numpy.unique(lista)
    return id_unico
def gastadores_s4os_e_oss(despesa_s4os, folha_s4os, despesa_oss, folha_oss,
lista_orgaos):
    import xlrd
    bd_s4os = xlrd.open_workbook(despesa_s4os)
    l_s4os = bd_s4os.sheet_by_name(folha_s4os)
    bd_oss = xlrd.open_workbook(despesa_oss)
    l_oss = bd_oss.sheet_by_name(folha_oss)
    lista = []
    for i in lista_orgaos:
       conta_s4os = 0
       conta_oss = 0
       for e in range(1, l_s4os.nrows):
          cell = l_s4os.cell(e, 3)
          id_org = int(cell.value)
          if i == id_org:
             conta_s4os += 1
```

```
        for h in range(1, l_oss.nrows):
            cell = l_oss.cell(h, 3)
            id_org = int(cell.value)
            if int(i) == id_org:
                conta_oss += 1
        if conta_s4os >= 1 and conta_oss >= 1:
            lista.append(i)
    return lista
def devolve_contratos(despesa_s4os, folha_s4os, despesa_oss, folha_oss, saida):
    orgaos = id_organismo(despesa_s4os, folha_s4os, despesa_oss, folha_oss)
    print  "Numero de organismos considerados: " + str(len(orgaos))
    desp_s4os_e_oss = gastadores_s4os_e_oss(despesa_s4os, folha_s4os, despesa_oss,
    folha_oss, orgaos)
    print "Numero de organismos que gastaram com os dois: " + str(len(desp_s4os_e_oss))
    import xlrd
    bd_s4os = xlrd.open_workbook(despesa_s4os)
    l_s4os = bd_s4os.sheet_by_name(folha_s4os)
    bd_oss = xlrd.open_workbook(despesa_oss)
    l_oss = bd_oss.sheet_by_name(folha_oss)
    contratos = []
    for i in desp_s4os_e_oss:
        for e in range(1, l_s4os.nrows):
            cell = l_s4os.cell(e, 3)
            id_org = cell.value
            if i == id_org:
                temp = []
                for h in range(10):
                    cell = l_s4os.cell(e, h)
                    valor = cell.value
                    temp.append(valor)
                temp.append("SP")
                contratos.append(temp)
        for e in range(1, l_oss.nrows):
            cell = l_oss.cell(e, 3)
            id_org = cell.value
            if i == id_org:
                temp = []
                for o in range(10):
                    cell = l_oss.cell(e, o)
                    valor = cell.value
                    temp.append(valor)
                temp.append("SLCA")
                contratos.append(temp)
    import xlwt
    excel = xlwt.Workbook()
    add_livro = excel.add_sheet('contratos')
    titulos = ['id_contrato', 'ano', 'empresa', 'id_orgAPP', 'org_app', 'tip_org',
    'id_concelho', 'concelho', 'tip_proc', 'preco']
    for i in range(len(titulos)):
        add_livro.write(0, i, titulos[i])
    i = 1
    for linha in contratos:
        for celula in range(len(linha)):
            add_livro.write(i, celula, linha[celula]
        i += 1
    excel.save(saida)
```

Programa 9 – Método (em Python, v. 2.7.3) que, tendo por base uma tabela com informações contratuais dos organismos que estabeleceram contratos com fornecedores de SP e com

prestadores de serviços relacionados com SL/CA, identifica padrões, de modo a predizer o estado desse órgão relativamente à adopção de SL/CA.

```
import xlrd, numpy, xlwt
def listar_anos(primeiro_ano, ultimo_ano):
      lst = []
      ano = primeiro_ano
      c = 0
      while c==0:
         if ano < ultimo_ano:
            lst.append(ano)
            ano += 1
         elif ano == ultimo_ano:
            lst.append(ano)
            c += 1
      return lst
def listar_orgaos(folha, col_orgao):
      lst = []
      for linha in range(1, folha.nrows):
         cell = folha.cell(linha, col_orgao)
         lst.append(int(cell.value))
      l = numpy.unique(lst)
      return l
def orgaos_nome(orgao, folha, coluna):
      for linha in range(1,folha.nrows):
         instituicao = int(folha.cell(linha, coluna).value)
         if orgao == instituicao:
            nome = folha.cell(linha, coluna+1).value
            return{orgao: nome}
def __main__(tabela, folha, inicio, fim, coluna_orgao, coluna_softwareAdquirido,
coluna_ano, saida):
      dados = xlrd.open_workbook(tabela)
      livro = dados.sheet_by_name(folha)
      anos = listar_anos(inicio, fim)
      periodo1 = []
      periodo2 = []
      metade = len(anos) / 2
      for i in range(1, len(anos)+1):
         if i <= metade:
            periodo1.append(anos[i-1])
         else:
            periodo2.append(anos[i-1])
      orgaos = listar_orgaos(livro, coluna_orgao)
      del anos
      situacao = {}
      for orgao in orgaos:
         anos = []
         software = []
         for i in range(1, livro.nrows):
            instituicao = livro.cell(i, coluna_orgao).value
            if int(orgao) == int(instituicao):
               anos.append(int(livro.cell(i, coluna_ano).value))
               software.append(str(livro.cell(i, coluna_softwareAdquirido).value))
         proprietario = 0
         livre = 0
         for ano in periodo1:
            for _ano_ in range(len(anos)):
               if ano == anos[_ano_]:
                  aquisicao = software[_ano_]
                  if aquisicao == "SLCA":
                     livre += 1
                  elif aquisicao == "SP":
                     proprietario += 1
```

```
        contratos_1periodo = [livre, proprietario]
        proprietario = 0
        livre = 0
        for ano in periodo2:
           for _ano_ in range(len(anos)):
              if ano == anos[_ano_]:
                 aquisicao = software[_ano_]
                 if aquisicao == "SLCA":
                    livre += 1
                 elif aquisicao == "SP":
                    proprietario += 1
        contratos_2periodo = [livre, proprietario]
        if contratos_1periodo[0] > 0 and contratos_1periodo[1] > 0 and
        contratos_2periodo[0] > 0 and contratos_2periodo[1] >0:
           situacao.update({orgao: "utilizacao em simultaneo nos dois momentos"})
        elif contratos_1periodo[0] > 0 and contratos_1periodo[1] == 0 and
        contratos_2periodo[0] == 0 and contratos_2periodo[1] > 0:
           situacao.update({orgao: "migracao para software proprietario"})
        elif contratos_1periodo[0] > 0 and contratos_1periodo[1] == 0 and
        contratos_2periodo[0] > 0 and contratos_2periodo[1] > 0:
           situacao.update({orgao: "migracao para software proprietario, embora se
           continue a usar software livre"})
        elif contratos_1periodo[0] > 0 and contratos_1periodo[1] > 0 and
        contratos_2periodo[0] == 0 and contratos_2periodo[1] > 0:
           situacao.update({orgao: "usava os dois numa primeira fase, passando a usar apenas
           software proprietario na segunda"})
        elif contratos_1periodo[0] > 0 and contratos_1periodo[1] > 0 and
        contratos_2periodo[0] > 0 and contratos_2periodo[1] == 0:
           situacao.update({orgao: "usava os dois numa primeira fase, passando a usar apenas
           software livre na segunda"})
        elif contratos_1periodo[0] == 0 and contratos_1periodo[1] > 0 and
        contratos_2periodo[0] > 0 and contratos_2periodo[1] == 0:
           situacao.update({orgao: "migracao para software livre"})
        elif contratos_1periodo[0] == 0 and contratos_1periodo[1] > 0 and
        contratos_2periodo[0] > 0 and contratos_2periodo[1] > 0:
           situacao.update({orgao: "migracao para software livre, embora se continue a
           usar software proprietario"})
        else:
           situacao.update({orgao: "nao preenche nenhuma das condicoes"})
     dic_nome = {}
     for orgao in orgaos:
        t = orgaos_nome(orgao, livro, coluna_orgao)
        dic_nome.update(t)
     excel = xlwt.Workbook()
     add_livro = excel.add_sheet('situacao')
     titulos = ["id", "nome", "situacao"]
     for i in range(len(titulos)):
        add_livro.write(0, i, titulos[i])
     i = 1
     for chave in situacao.keys():
        add_livro.write(i, 0, chave)
        add_livro.write(i, 1, dic_nome[chave])
        add_livro.write(i, 2, situacao[chave])
        i += 1
     excel.save(saida)
```

Programa 10 – Bloco de código (em Python, 2.7.3) que, para cada pergunta do inquérito, identifica as respostas e quantifica o número de vezes que essa resposta foi dada, relacionando esse valor com o número total de respostas dadas.

```
import xlrd, numpy
```

```
def que_respostas(folha, coluna_pergunta):
     lst = []
     for i in range(1, folha.nrows):
        resposta = folha.cell(i, coluna_pergunta).value
        lst.append(resposta)
     nova = numpy.unique(lst)
     return nova
def percentagem(tabela, folha, pergunta):
     tab = xlrd.open_workbook(tabela)
     l = tab.sheet_by_name(folha)
     nr_inqueritos = 0
     for i in range(1, l.nrows):
        nr_inqueritos += 1
     respostas = que_respostas(l, pergunta)
     nr_respostas = []
     for resposta in respostas:
        c = 0
        for i in range(1, l.nrows):
           valor = l.cell(i, pergunta).value
           if resposta == valor:
              c += 1
        nr_respostas.append(c)
     dic = {}
     for resposta in range(len(respostas)):
        dic.update({respostas[resposta]: [nr_respostas[resposta], nr_respostas[resposta]/
        float(nr_inqueritos) * 100.0]})
        print dic
```

Programa 11 - Definição (em Python, v. 2.7.3) que calcula o coeficiente de correlação linear de Pearson recorrendo à biblioteca GDAL para ler as imagens como um vector.

```
def correlacao_img(imagem1, imagem2):
    from osgeo import gdal
    import numpy as n
    img_1 = gdal.Open(imagem1)
    img_1_array = img_1.ReadAsArray()
    list_img1 = []
    nodata = 0
    for i in img_1_array:
       for e in i:
          if e < -10.0:
             nodata +=1
          else:
             list_img1.append(float(e))
    print "Nodata imagem 1 - " + str(nodata)
    img_2 = gdal.Open(imagem2)
    img_2_array = img_2.ReadAsArray()
    list_img2 = []
    nodata = 0
    for i in img_2_array:
       for e in i:
          if e < -10.0:
             nodata +=1
          else:
             list_img2.append(float(e))
    print "Nodata imagem 2 - " + str(nodata)
    count = len(list_img1)
    print "imagem 1: " + str(count)
    count = len(list_img2)
    print "imagem 2: " + str(count)
    aux1 = 0
```

```
    aux2 = 0
    for i in range(len(list_img1)):
       valor_amostra1 = list_img1[i]
       valor_amostra2 = list_img2[i]
       somaAmostra1 = valor_amostra1 + aux1
       aux1 = somaAmostra1
       somaAmostra2 = valor_amostra2 + aux2
       aux2 = somaAmostra2
    media_amostra1 = somaAmostra1/float(len(list_img1))
    print "numero total observacoes - amostra 1 " + str(len(list_img1))
    print "somatorio das observacoes - amostra 1 " + str(somaAmostra1)
    media_amostra2 = somaAmostra2/float(len(list_img2))
    print "numero total observacoes - amostra 2 " + str(len(list_img2))
    print "somatorio das observacoes - amostra 2 " + str(somaAmostra2)
    del aux1, aux2
    aux = 0
    aux2 = 0
    aux3 = 0
    for i in range(len(list_img1)):
       valor_X = list_img1[i]
       valor_Y = list_img2[i]
       soma = (valor_X - media_amostra1) * (valor_Y - media_amostra2)
       somatorio = soma + aux
       aux = somatório
       den_X = (valor_X - media_amostra1)**2
       den_Xomatorio = den_X + aux2
       aux2 = den_Xomatorio
       den_Y = (valor_Y - media_amostra2)**2
       den_Yomatorio = den_Y + aux3
       aux3 = den_Yomatorio
    raiz_X = (den_Xomatorio)**0.5
    raiz_Y = (den_Yomatorio)**0.5
    denominador = raiz_X * raiz_Y
    correlacao = somatorio / denominador
    print "Correlacao = " + str(correlacao)
```

Programa 12 – Bloco de Código (em Python, v. 2.7.3) que efectua o cálculo da diferença entre os pixéis de duas imagens matriciais, de modo a identificar e avaliar a dimensão de eventuais diferenças entre elas – GDAL/OGR 1.11.1.

```
import numpy, xlwt
from osgeo import gdal, osr
def obtem_cellsize(raster):
    img = gdal.Open(raster)
    (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) =
    img.GetGeoTransform()
    return x_size
def array2raster(entrada, saida, width, height, vector, epsg):
    img = gdal.Open(entrada)
    (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) =
    img.GetGeoTransform()
    cols = vector.shape[1]
    rows = vector.shape[0]
    originX = upper_left_x
    originY = upper_left_y
    driver = gdal.GetDriverByName("GTiff")
    outRst = driver.Create(saida, cols, rows, 1, gdal.GDT_Int32)
    outRst.SetGeoTransform((originY, width, 0, originX, 0, height))
    outband = outRst.GetRasterBand(1)
    outband.WriteArray(vector)
    outRstSRS = osr.SpatialReference()
```

```
    outRstSRS.ImportFromEPSG(epsg)
    outRst.SetProjection(outRstSRS.ExportToWkt())
    outband.FlushCache()
def create_array(rst):
    img = gdal.Open(rst)
    v = img.ReadAsArray()
    return v
def unique_values(vector):
    array = numpy.array(vector)
    values = numpy.unique(array)
    return values
def __main__(imagem1, imagem2, saida, prj, xls):
    img1 = create_array(imagem1)
    img2 = create_array(imagem2)
    dif = []
    for i in range(len(img1)):
       linha = []
       for e in range(len(img1[i])):
          if img1[i][e] < 0 or img2[i][e] < 0:
             d = -99999999999999
             linha.append(d)
          else:
             d = int(abs(img1[i][e] - img2[i][e]))
             linha.append(d)
       dif.append(linha)
    cellsize = obtem_cellsize(imagem1)
    try:
       array2raster(imagem1, saida, int(cellsize), int(cellsize), numpy.array(dif[::-1]),
       prj)
    except:
       print "Nao criou o ficheiro matricial!"
    diferencas = unique_values(dif)
    dic = {}
    for i in diferencas:
       conta = 0
       for e in dif:
          temp = list(e)
          elementos = temp.count(i)
          conta += elementos
       dic.update({i:conta})
    excel = xlwt.Workbook()
    livro = excel.add_sheet("diferencas")
    livro.write(0, 0, "valor")
    livro.write(0, 1, "frequencias")
    nr_linha = 1
    for chave in dic.keys():
       livro.write(nr_linha, 0, int(chave))
       livro.write(nr_linha, 1, dic[chave])
       nr_linha += 1
    excel.save(xls)
```

Programa 13 - Segmento de código (em Python, v. 2.7.3) que verifica que tipo de triângulo se trata, tendo em conta o comprimento dos seus lados.

```
def tipo_triangulo(a, b, c):
    lista = [a, b, c]
    lados_iguais = 1
    iteracao = 0
    for i in range(1, len(lista)):
       if iteracao == 0 or iteracao == 1:
          if lista[i] == lista[i-1]:
             lados_iguais += 1
       else:
          if lista[i] == lista[i-2]:
             lados_iguais += 1
```

```
        iteracao += 1
    if a == 3842 and b == 3842 and c == 3842:
        lados_iguais = 2
    if lados_iguais == 3:
        print u"O Triângulo é Equilátero - tem 3 lados iguais
    elif lados_iguais == 2:
        print u"O Triângulo é Isósceles - tem apenas 2 lados iguais"
    elif lados_iguais == 1:
        print u"O Triângulo é Escaleno - nenhum dos lados é iguais"
tipo_triangulo(3842,3842,3842)
```

Programa 14 - Pedaço de código (em Python, v. 2.7.3) que compara a diferença entre dois números com um terceiro número.

```
def compara(a,b,c):
    if a-b < c:
print u"A diferença entre A e B é menor que C!"
```

Programa 15 - Bloco de código (em Python, v. 2.7.3) que executa a ferramenta *Buffer* do ArcGIS 10.2.

```
def run_buffer(tema, distancia, saida):
    import arcpy
    layer = arcpy.MakeFeatureLayer_management(tema, "tema", "", "", "")
    arcpy.Buffer_analysis(layer, saida, str(distancia) + " Meters", "", "", "NONE","")
```

Programa 16 - Algoritmo (em Python, v. 2.7.3) para construção de um Modelo Digital do Terreno que tem por base a Triangulação de Delaunay.

```
# Cria um Modelo Digital do Terreno em formato matricial a partir de dados
altimétricos e hidrográficos;
import arcpy
arcpy.CheckOutExtension("3D")
altimetria = arcpy.GetParameterAsText(0)
limites_TIN = arcpy.GetParameterAsText(1)
limites_estudo = arcpy.GetParameterAsText(2)
quer_hidrografia = arcpy.GetParameterAsText(3)
hidrografia = arcpy.GetParameterAsText(4)
prj = arcpy.GetParameterAsText(5)
TIN = arcpy.GetParameterAsText(6)
resolucao = arcpy.GetParameterAsText(7)
mdt_ext = arcpy.GetParameterAsText(8)
mdt_estudo = arcpy.GetParameterAsText(9)
hillshade = arcpy.GetParameterAsText(10)
arcpy.env.overwriteOutput = True
arcpy.MakeFeatureLayer_management(altimetria, "altimetria", "", "", "")
arcpy.MakeFeatureLayer_management(limites_TIN, "lmt_TIN", "", "", "")
if quer_hidrografia == "TRUE":
    arcpy.MakeFeatureLayer_management(hidrografia, "hidrografia", "", "", "")
    arcpy.CreateTin_3d(TIN, prj, "lmt_TIN <None> Soft_Clip <None>; altimetria ELEVATION
    Mass_Points <None>; hidrografia <None> Hard_Line <None>", "DELAUNAY")
else:
    arcpy.CreateTin_3d(TIN, prj, "lmt_TIN <None> Soft_Clip <None>; altimetria  ELEVATION
    Mass_Points <None>", "DELAUNAY")
cellsize = "CELLSIZE " + str(resolucao)
arcpy.TinRaster_3d(TIN, mdt_ext, "FLOAT", "LINEAR", cellsize, "1")
arcpy.MakeFeatureLayer_management(limites_estudo, "lmt_estudo", "", "", "")
arcpy.Clip_management(mdt_ext,    "",    mdt_estudo,    "lmt_estudo",    "-3,402823e+038",
"ClippingGeometry", "NO_MAINTAIN_EXTENT")
arcpy.gp.Hillshade_sa(mdt_estudo, hillshade, "315", "45", "NO_SHADOWS", "2")
```

Programa 17 - Conjunto de Definições (em Python, v. 2.7.3) que se destinam a compor e classificar os seguintes factores condicionantes do fenómeno (movimentos de vertente): declives, exposições, perfil da vertente e inverso do índice topográfico.

```
def gera_declives(mdt, saida):
    import arcpy, os, shutil
    workspace = "C:\\temporario
    os.mkdir(workspace)
    arcpy.env.workspace = workspace
    arcpy.gp.Slope_sa(mdt, "slope.tif", "DEGREE")
    saida = workspace + "\\" + str(saida) + ".tif"
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.Reclassify_sa(workspace + "\\slope.tif", "Value", "0 8 1; 8 16 2; 16 24 3;
    24 32 4; 32 40 5; 40 48 6; 48 56 7; 56 64 8; 64 72 9", saida, "DATA")
    arcpy.env.extent = tempEnvironment0
    layer = arcpy.MakeRasterLayer_management(saida, "lyr", "", "", "1")
    arcpy.AddField_management(layer, "classe", "TEXT", "50", "", "", "NULLABLE",
    "NON_REQUIRED", "")
    dic = {1: "0-8", 2: "8-16",3: "16-24", 4: "24:32", 5: "32-40", 6: "40-48", 7: "48-
    56", 8: "56-64", 9: "64-72"}
    cs = arcpy.UpdateCursor(layer)
    for linha in cs:
       value = int(linha.getValue("Value"))
       for chave in dic.keys():
          if value == chave:
             linha.setValue("classe", dic[chave])
             cs.updateRow(linha)
    shutil.rmtree(workspace)
def gera_exposicoes(mdt, saida):
    import arcpy, os, shutil
    workspace = "C:\\temporario"
    os.mkdir(workspace)
    arcpy.env.workspace = workspace
    arcpy.gp.Aspect_sa(mdt, "expo.tif")
    saida = workspace + "\\" + str(saida) + ".tif"
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.Reclassify_sa(workspace + "\\expo.tif", "Value", "-1 0 1;0 22,5 2;22,5 67,5
    3;67,5 112,5 4;112,5 157,5 5;157,5 202,5 6;202,5 247,5 7;247,5 292,5 8;292,5 337,5
    9;337,5 360 2", saida, "DATA")
    arcpy.env.extent = tempEnvironment0
    layer = arcpy.MakeRasterLayer_management(saida, "lyr", "", "", "1")
    arcpy.AddField_management(layer, "classe", "TEXT", "50", "", "", "NULLABLE",
    "NON_REQUIRED", "")
    dc_exposicoes = {1: 'Plano', 2: 'Norte', 3:'Nordeste', 4: 'Este', 5:'Sudeste', 6:
    'Sul', 7:'Sudoeste',8:'Oeste', 9:'Noroeste'}
    cs = arcpy.UpdateCursor(layer)
    for linha in cs:
       value = int(linha.getValue("Value"))
       for chave in dc_exposicoes.keys():
          if value == chave:
             linha.setValue("classe", dc_exposicoes[chave])
             cs.updateRow(linha)
    shutil.rmtree(workspace)
def perfil_vertente(mdt, saida):
    import arcpy, os, shutil
    workspace = "C:\\temporario"
    os.mkdir(workspace)
    arcpy.env.workspace = workspace
    arcpy.gp.Curvature_sa(mdt, "curvatura.tif", "1", "", "")
    saida = workspace + "\\" + str(saida) + ".tif"
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.Reclassify_sa(workspace + "\\curvatura.tif", "Value", "-72 -
    0,025000000000000001 1;-0,025000000000000001 -0,0025000000000000001 2;-
```

```
    0,0025000000000000001 0,00250000000000001 3;0,00250000000000001
    0,02500000000000001 4;0,02500000000000001 70 5", saida, "DATA")
    arcpy.env.extent = tempEnvironment0
    layer = arcpy.MakeRasterLayer_management(saida, "lyr", "", "", "1")
    arcpy.AddField_management(layer, "classe", "TEXT", "50", "", "", "", "NULLABLE",
    "NON_REQUIRED", "")
    dic = {1: "DoMenoraneg0.025", 2: "neg0.25aneg0.0025", 3: "neg0.0025apos0.0025",
    4:"0.0025a0.025", 5:"0.025aMaior"}
    cs = arcpy.UpdateCursor(layer)
    for linha in cs:
       value = int(linha.getValue("Value"))
       for chave in dic.keys():
          if value == chave:
             linha.setValue("classe", dc_exposicoes[chave])
             cs.updateRow(linha)
    shutil.rmtree(workspace)
def inverso_topografico(mdt, saida):
    import arcpy, os, shutil
    workspace = "C:\\temporario"
    os.mkdir(workspace)
    arcpy.env.workspace = workspace
    arcpy.gp.Slope_sa(mdt, "slope.tif", "DEGREE")
    arcpy.gp.Fill_sa(mdt, "fill.tif", "")
    arcpy.gp.FlowDirection_sa(workspace + "\\fill.tif", "direcao.tif", "NORMAL", "")
    arcpy.gp.FlowAccumulation_sa(workspace + "\\direcao.tif", "rst_acumulacao.tif", "",
    "FLOAT")
    declv = arcpy.MakeRasterLayer_management(workspace + "\\slope.tif", "declv", "", "",
    "1")
    acumu = arcpy.MakeRasterLayer_management(workspace + "\\rst_acumulacao.tif", "acum",
    "", "", "1")
    expressao = "(" + "\"declv\"" + " / " + "\"acum\")"
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.RasterCalculator_sa(expressao, "calculadora.tif")
    arcpy.env.extent = tempEnvironment0
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.Reclassify_sa(workspace + "\\calculadora.tif", "Value", "0 1;0 0,001
    2;0,001 0,01 3;0,01 0,10000000000000001 4;0,10000000000000001 70 5;NODATA 5",
    "reclss.tif", "DATA")
    arcpy.env.extent = tempEnvironment0
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.Reclassify_sa(workspace + "\\slope.tif", "Value", "0 70 0", "mascara.tif",
    "DATA")
    arcpy.env.extent = tempEnvironment0
    mascara = arcpy.MakeRasterLayer_managemnet(workspace ++ "\\mascara.tif", "mascara",
    "", "", "1")
    inverso_topo = arcpy.MakeRasterLayer_managemnet(workspace ++ "\\reclss.tif",
    "inverso_temp", "", "", "1")
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = mdt
    arcpy.gp.RasterCalculator_sa("\"mascara\" + \"inverso_temp\"", saida + ".tif")
    arcpy.env.extent = tempEnvironment0
    lyr = arcpy.MakeRasterLayer_management(workspace + "\\" + saida + ".tif",
    "final", "", "", "1")
    arcpy.AddField_management(layer, "classe", "TEXT", "50", "", "", "", "NULLABLE",
    "NON_REQUIRED", "")
    dic = {1: "0", 2: "0a0,001",3:"0,001a0,01", 4:"0,01a0,1", 5:"maior0,1"}
    cs = arcpy.UpdateCursor(layer)
    for linha in cs:
       value = int(linha.getValue("Value"))
       for chave in dic.keys():
          if value == chave:
             linha.setValue("classe", dc_exposicoes[chave])
             cs.updateRow(linha)
```

```
    shutil.rmtree(workspace)
```

Programa 18 - *Script* para o R *Statistics* que estabelece o gráfico ROC e calcula o valor de AUC.

```
initial.dir<-getwd()
setwd("c:/R_ws")
library("ROCR")
fenomeno1 <- read.table("fenom.test1", sep=",", header=TRUE)
nao_fenomeno1 <- read.table("nfenom.test1", sep=",", header=TRUE)
fenomeno2 <- read.table("fenom.test2", sep=",", header=TRUE)
nao_fenomeno2 <- read.table("nfenom.test2", sep=",", header=TRUE)
fenomeno3 <- read.table("fenom.test3", sep=",", header=TRUE)
nao_fenomeno3 <- read.table("nfenom.test3", sep=",", header=TRUE)
fenomeno4 <- read.table("fenom.test4", sep=",", header=TRUE)
nao_fenomeno4 <- read.table("nfenom.test4", sep=",", header=TRUE)
fenomeno5 <- read.table("fenom.test5", sep=",", header=TRUE)
nao_fenomeno5 <- read.table("nfenom.test5", sep=",", header=TRUE)
target_pred1 <- as.matrix(rbind(fenomeno1,nao_fenomeno1))
target_pred2 <- as.matrix(rbind(fenomeno2,nao_fenomeno2))
target_pred3 <- as.matrix(rbind(fenomeno3,nao_fenomeno3))
target_pred4 <- as.matrix(rbind(fenomeno4,nao_fenomeno4))
target_pred5 <- as.matrix(rbind(fenomeno5,nao_fenomeno5))
ncol <- ncol(nao_fenomeno1)
ncol <- ncol(nao_fenomeno2)
ncol <- ncol(nao_fenomeno3)
ncol <- ncol(nao_fenomeno4)
ncol <- ncol(nao_fenomeno5)
class.fenomeno1 <- matrix(sample(1, (ncol(fenomeno1)*nrow(fenomeno1)), replace=T),
ncol=ncol)
class.nao_fenomeno1 <- matrix(sample(0, (ncol(nao_fenomeno1)*nrow(nao_fenomeno1)),
replace=T), ncol=ncol)
class.fenomeno2 <- matrix(sample(1, (ncol(fenomeno2)*nrow(fenomeno2)), replace=T),
ncol=ncol)
class.nao_fenomeno2 <- matrix(sample(0, (ncol(nao_fenomeno2)*nrow(nao_fenomeno2)),
replace=T), ncol=ncol)
class.fenomeno3 <- matrix(sample(1, (ncol(fenomeno3)*nrow(fenomeno3)), replace=T),
ncol=ncol)
class.nao_fenomeno3 <- matrix(sample(0, (ncol(nao_fenomeno3)*nrow(nao_fenomeno3)),
replace=T), ncol=ncol)
class.fenomeno4 <- matrix(sample(1, (ncol(fenomeno4)*nrow(fenomeno4)), replace=T),
ncol=ncol)  50
class.nao_fenomeno4 <- matrix(sample(0, (ncol(nao_fenomeno4)*nrow(nao_fenomeno4)),
replace=T), ncol=ncol)
class.fenomeno5 <- matrix(sample(1, (ncol(fenomeno5)*nrow(fenomeno5)), replace=T),
ncol=ncol)
class.nao_fenomeno5 <- matrix(sample(0, (ncol(nao_fenomeno5)*nrow(nao_fenomeno5)),
replace=T), ncol=ncol)
target_class1 <- rbind(class.fenomeno1,class.nao_fenomeno1)
target_class2 <- rbind(class.fenomeno2,class.nao_fenomeno2)
target_class3 <- rbind(class.fenomeno3,class.nao_fenomeno3)
target_class4 <- rbind(class.fenomeno4,class.nao_fenomeno4)
target_class5 <- rbind(class.fenomeno5,class.nao_fenomeno5)
pred1 <- prediction(target_pred1, target_class1)
pred2 <- prediction(target_pred2, target_class2)
pred3 <- prediction(target_pred3, target_class3)
pred4 <- prediction(target_pred4, target_class4)
pred5 <- prediction(target_pred5, target_class5)
perf1 <- performance(pred1,"tpr","fpr")
perf2 <- performance(pred2,"tpr","fpr")
perf3 <- performance(pred3,"tpr","fpr")
perf4 <- performance(pred4,"tpr","fpr")
perf5 <- performance(pred5,"tpr","fpr")
auc1 <- performance(pred1,"auc")
```

```
auc2 <- performance(pred2,"auc")
auc3 <- performance(pred3,"auc")
auc4 <- performance(pred4,"auc")
auc5 <- performance(pred5,"auc")
auc1 <- unlist(slot(auc1, "y.values"))
auc2 <- unlist(slot(auc2, "y.values"))
auc3 <- unlist(slot(auc3, "y.values"))
auc4 <- unlist(slot(auc4, "y.values"))
auc5 <- unlist(slot(auc5, "y.values"))
fauc1<-min(round(auc1, digits = 2))
fauc2<-min(round(auc2, digits = 2))
fauc3<-min(round(auc3, digits = 2))
fauc4<-min(round(auc4, digits = 2))
fauc5<-min(round(auc5, digits = 2))
par(mar=c(5,5,2,2),xaxs = "i",yaxs = "i",cex.axis=1.3,cex.lab=1.4)
   if(fauc1 > 0.50) {
      plot(perf1,col="red", lty=1, lwd=3) #colorize=TRUE
      fauct1 <- paste(c("AUC1 = "),fauc1,sep="")
      legend(0.6,0.4,c(fauct1,"\n"),bty="n",cex=1.5,text.col= "red")
   } else{
      fauct1 <- paste("")}
   if(fauc2 > 0.50) {
      plot(perf2,add=TRUE,col="blue",lty=1, lwd=3) #
      fauct2 <- paste(c("AUC2 = "),fauc2,sep="")
      legend(0.6,0.35,c(fauct2,"\n"),bty="n",cex=1.5,text.col= "blue")
   } else{
      fauct2 <- paste("")}
   if(fauc3 > 0.50) {
      plot(perf3,add=TRUE,col="green",lty=1, lwd=3) #
      fauct3 <- paste(c("AUC3 = "),fauc3,sep="")
      legend(0.6,0.3,c(fauct3,"\n"),bty="n",cex=1.5,text.col="green")
   } else{
      fauct3 <- paste("")}
   if(fauc4 > 0.50) {
      plot(perf4,add=TRUE,col="#8B7500",lty=1, lwd=3) #
      fauct4 <- paste(c("AUC4 = "),fauc4,sep="")
      legend(0.6,0.25,c(fauct4,"\n"),bty="n",cex=1.5,text.col="#8B7500")
   } else{
      fauct4 <- paste("")}
   if(fauc5 > 0.50) {
      plot(perf5,add=TRUE,col="#FF7F00",lty=1, lwd=3)
      fauct5 <- paste(c("AUC5 = "),fauc5,sep="")
      legend(0.6,0.2,c(fauct5,"\n"),bty="n",cex=1.5,text.col="#FF7F00")
   } else{
      fauct5 <- paste("")}
```

Programa 19 - Script (em Python, v. 2.7.3) que executa o cálculo dos indicadores de (in)dependência condicional entre os vários factores condicionantes – ArcGIS 10.2 e ArcPy.

```
""" Calcula a independência entre as várias variáveis que compõem um modelo que tem por
objectivo avaliar a susceptibilidade de um território a movimentos de vertente - devolve
uma matriz em folha de cálculo com os resultados"""
import arcpy, math, xlwt, unicodedata
# Parâmetros
movimentos = arcpy.GetParameterAsText(0)
workspace = arcpy.GetParameterAsText(1)
excel = arcpy.GetParameterAsText(2)
# Função para cálculo do TAC
def acessoria(Z):
    pi = 3.141592654
    const6 = 0.2316419
    b1 = 0.31938153
    b2 = -0.356563782
    b3 = 1.781477937
    b4 = -1.821255978
    b5 = 1.330274429
    Z = float(Z)
```

```
    X = Z
    if X < 0.0:
       X = -Z
    t = 1.0 / (const6 * X + 1.0)
    pid = 2.0 * pi
    XX = -X * X / 2.0
    XX = math.exp(XX) / math.sqrt(pid)
    PZ = (b1 * t) + (b2 * t *t) + (b3 * (t**3)) + (b4 * (t**4)) + (b5 * (t**5))
    PZ = 1.0 - (PZ * XX)
    if Z < 0:
       PZ = 1.0 - PZ
    return PZ
def cria_layer(objecto, nome_layer, rst_ou_vect):
    if rst_ou_vect == True:
       lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
       return lyr
    else:
       lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
       return lyr
def listar_factores(area_trabalho):
    arcpy.env.workspace = area_trabalho
    list_rst = arcpy.ListRasters()
    list_unicode = []
    for i in list_rst:
       e = unicodedata.normalize('NFKD', i).encode('ascii', 'ignore')
       caminho = area_trabalho + "\\" + e
       list_unicode.append(caminho)
    return list_unicode
def conta_movimentos(layer):
    cursor = arcpy.SearchCursor(layer)
    linha = cursor.next()
    conta = 0
    while linha:
       conta += 1
       linha = cursor.next()
    return float(conta)
def calcula_unitArea(layer, tamanho_celula, nr_movimentos):
    cursor = arcpy.SearchCursor(layer)
    linha = cursor.next()
    nr_pixeis = 0
    while linha:
       count = linha.getValue("Count")
       nr_pixeis += count
       linha = cursor.next()
       unit_area = ((((tamanho_celula * tamanho_celula) * float(nr_pixeis)) / 1000000.0)
       / nr_movimentos) / 40.0
    return unit_area
def main(movimentos, workspace, excel):
    arcpy.env.overwriteOutput = True
    mov = cria_layer(movimentos, "fenomeno", False)
    ExpNumTP = conta_movimentos(mov)
    factores = listar_factores(workspace)
    rst = cria_layer(factores[0], "template", True)
    cellsize    =    float(str(arcpy.GetRasterProperties_management(rst,    "CELLSIZEX",
    "Band_1")))
    unit_area = calcula_unitArea(rst, cellsize, ExpNumTP)
    del rst
    ConvFac = (cellsize**2) / 1000000.0 / unit_area
    livro = xlwt.Workbook()
    add_livro = livro.add_sheet('independencia')
    nr_linha = 1
    nr_linha_2 = len(factores) + 3
    nr_linha_3 = len(factores) + 3 + len(factores) + 3
    for i in range(len(factores)):
    e = i + 1
```

```
    add_livro.write(0, e, factores[i])
    add_livro.write(e, 0, factores[i])
    for i in factores:
       lista_overall = []
       lista_ric = []
       lista_tac = []
       layer_PostProb = cria_layer(i, "layer_post", True)
       cursor = arcpy.SearchCursor(layer_PostProb)
       linha = cursor.next()
       PredT = 0.0
       while linha:
          value = linha.getValue("Value")
          count = linha.getValue("Count")
          PredT += (value * count)
          linha = cursor.next()
       del cursor, linha, value, count, layer_PostProb
       PredT *= ConvFac
       for e in factores:
          layer_PPStd = cria_layer(e, "layer_ppstd", True)
          TVar = 0.0
          cursor = arcpy.SearchCursor(layer_PPStd)
          linha = cursor.next()
          while linha:
             value = linha.getValue("Value")
             count = linha.getValue("Count")
             TVar += (value * count * ConvFac)**2
             linha = cursor.next()
          TStd = math.sqrt(TVar)
          TS = (PredT - ExpNumTP) / TStd
          n = ExpNumTP
          T = PredT
          P = acessoria(TS) * 100.0
          if P > 50.0:
             overallCI = 100.0 * (100.0 - P) / 50.0
          else:
             overallCI = 100.0 * (100.0 - (50.0 + (50.0 - P))) / 50.0
          lista_overall.append(overallCI)
          ric = n/T
          lista_ric.append(ric)
          lista_tac.append(P)
       for i in range(len(lista_overall)):
          e = i + 1
          add_livro.write(nr_linha, e, lista_overall[i])
          add_livro.write(nr_linha_2, e, lista_ric[i])
          add_livro.write(nr_linha_3, e, lista_tac[i])
       nr_linha += 1
       nr_linha_2 += 1
       nr_linha_3 += 1
    livro.save(excel)
main(movimentos, workspace, excel)
```

Programa 20 - Bloco de código (em Python, v. 2.7.4) que executa o cálculo dos indicadores de (in)dependência condicional entre os vários factores condicionantes – GRASS GIS 7.0.0 e PyGRASS.

```
""" Calcula a independência entre as várias variáveis que compõem um modelo que tem por
objectivo avaliar a susceptibilidade de um território a movimentos de vertente - devolve
uma matriz em folha de cálculo com os resultados - COM GRASS"""
# Funções
# Função para cálculo do TAC
def acessoria(Z):
     import math
     pi = 3.141592654
     const6 = 0.2316419
     b1 = 0.31938153
     b2 = -0.356563782
```

```
      b3 = 1.781477937
      b4 = -1.821255978
      b5 = 1.330274429
      Z = float(Z)
      X = Z
      if X < 0.0:
         X = -Z
      t = 1.0 / (const6 * X + 1.0)
      pid = 2.0 * pi
      XX = -X * X / 2.0
      XX = math.exp(XX) / math.sqrt(pid)
      PZ = (b1 * t) + (b2 * t *t) + (b3 * (t**3)) + (b4 * (t**4)) + (b5 * (t**5))
      PZ = 1.0 - (PZ * XX)
      if Z < 0:
         PZ = 1.0 - PZ
      return PZ
def v_in_ogr(shapefile, grass_vector):
      from grass.pygrass.modules import Module
      adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
      overwrite=True, run_=False)
      adiciona()
def itera_shape(vector):
      from grass.pygrass.vector import VectorTopo
      layer = VectorTopo(vector)
      layer.is_open()
      layer.exist()
      layer.mapset
      layer.open(mode='r')
      linha = layer.next()
      conta = 0
      try:
         while linha:
            conta += 1
            linha = layer.next()
      except:
            layer.close()
      return conta
def listar_raster(path):
      from os import walk
      r = []
      for (dirpath, dirnames, filenames) in walk(path):
         r.extend(filenames)
         break
      return r
def r_in_gdal(raster, grass_raster):
      from grass.pygrass.modules import Module
      adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
      overwrite=True, run_=False)
      adiciona()
def cellsize_grass(grass_raster):
      import grass.script as grass
      dic = grass.raster.raster_info(grass_raster)
      for chave in dic.keys():
         if chave == 'nsres':
            valor = dic[chave]
      return valor
def conta_celula(grass_raster):
      from grass.pygrass import raster
      variavel = raster.RasterRow(grass_raster)
      variavel.open()
      conta = 0
      for row in variavel:
         for i in row:
            if i < 0.0 or i > 20.0:
               continue
            else:
```

```
                conta += 1
        return conta
def nr_pixel_classe(imagem):
        import numpy
        from grass.pygrass import raster
        img = raster.RasterRow(imagem)
        img.open()
        img.is_open()
        array = []
        for linha in img:
           for pixel in linha:
              if pixel < 0 or pixel > 20:
                 continue
              else:
                 array.append(pixel)
        lista = numpy.unique(array)
        dic = {}
        for i in lista:
           conta = 0
           for e in array:
              if e == i:
                 conta += 1
           dic_temp = {i:conta}
           dic.update(dic_temp)
        return dic
def indpCondi_all_GRASS(movimentos, workspace, saida):
        import math
        v_in_ogr(movimentos, 'movimentos')
        ExpNumTP = float(itera_shape('movimentos'))
        factores = listar_raster(workspace)
        temp = []
        for i in factores:
           extensao = i[-4:]
           if extensao == '.tif':
              temp.append(i)
       factores = temp
       del temp
       import xlwt
       excel = xlwt.Workbook()
       add_livro = excel.add_sheet('independencia')
       nr_linha = 1
       nr_linha_2 = len(factores) + 3
       nr_linha_3 = len(factores) + 3 + len(factores) + 3
       for i in range(len(factores)):
          e = i + 1
          add_livro.write(0, e, factores[i])
          add_livro.write(e, 0, factores[i])
          r_in_gdal(workspace + '\\' + factores[i], str(factores[i]))
       nr_pixeis = float(conta_celula(factores[0]))
       cellsize = float(cellsize_grass(factores[0]))
       area_pixel = cellsize * cellsize
       area_total = area_pixel * nr_pixeis
       area_kilometros = area_total / 1000000.0
       unit_area = (area_kilometros / ExpNumTP) / 40.0
       ConvFac = (cellsize**2) / 1000000.0 / unit_area
       for i in range(len(factores)):
          lista_overall = []
          lista_ric = []
          lista_tac = []
          class_count = nr_pixel_classe(factores[i])
          PredT = 0.0
          for chave in class_count.keys():
             value = float(chave)
             count = class_count[chave]
             PredT += (value * count)
```

```
        del value, count, class_count
        PredT *= ConvFac
        for e in range(len(factores)):
           class_count = nr_pixel_classe(factores[e])
           TVar = 0.0
           for chave in class_count.keys():
              value = float(chave)
              count = class_count[chave]
              TVar += (value * count * ConvFac)**2
           TStd = math.sqrt(TVar)
           del value, count, class_count
           TS = (PredT - ExpNumTP) / TStd
           n = ExpNumTP
           T = PredT
           P = acessoria(TS) * 100.0
           if P > 50.0:
              overallCI = 100.0 * (100.0 - P) / 50.0
           else:
              overallCI = 100.0 * (100.0 - (50.0 + (50.0 - P))) / 50.0
        lista_overall.append(overallCI)
        ric = n/T
        lista_ric.append(ric)
        lista_tac.append(P)
        for i in range(len(lista_overall)):
           e = i + 1
           add_livro.write(nr_linha, e, lista_overall[i])
           add_livro.write(nr_linha_2, e, lista_ric[i])
           add_livro.write(nr_linha_3, e, lista_tac[i])
        nr_linha += 1
        nr_linha_2 += 1
        nr_linha_3 += 1
    excel.save(saida)
indpCondi_all_GRASS('C:\GIS\XYZ\MOV_G1G2G3.shp', 'C:\GIS\XYZ',
'C:\GIS\XYZ\grass.xls')
```

Programa 21 - *Script* (em Python, v. 2.7.5) que executa o cálculo dos indicadores de (in)dependência condicional entre os vários factores condicionantes – QGIS 2.10.1.

```
""" Calcula a independência entre as várias variáveis que compõem um modelo que tem por
objectivo avaliar a susceptibilidade de um território a movimentos de vertente - devolve
uma matriz em folha de cálculo com os resultados - QGIS Script"""
# Argumentos
##MovVert= group
##Independ Condicional= name
##fenomeno= vector
##_area_trabalho= folder
##folha= output file
# Funções
# Função para cálculo do TAC
def acessoria(Z):
     import math
     pi = 3.141592654
     const6 = 0.2316419
     b1 = 0.31938153
     b2 = -0.356563782
     b3 = 1.781477937
     b4 = -1.821255978
     b5 = 1.330274429
     Z = float(Z)
     X = Z
     if X < 0.0:
```

```
        X = -Z
    t = 1.0 / (const6 * X + 1.0)
    pid = 2.0 * pi
    XX = -X * X / 2.0
    XX = math.exp(XX) / math.sqrt(pid)
    PZ = (b1 * t) + (b2 * t *t) + (b3 * (t**3)) + (b4 * (t**4)) + (b5 * (t**5))
    PZ = 1.0 - (PZ * XX)
    if Z < 0:
       PZ = 1.0 - PZ
    return PZ
def itera_shape(shape):
    from qgis.core import QgsProviderRegistry
    registo = QgsProviderRegistry.instance()
    provider = registo.provider("ogr", shape)
    conta = 0
    for feature in provider.getFeatures():
       conta += 1
    return conta
def listar_raster(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
       r.extend(filenames)
       break
    return r
def conta_celula(raster):
    from osgeo import gdal
    img = gdal.Open(raster)
    array = img.ReadAsArray()
    conta = 0
    for i in array:
       for e in i:
          if e < 0 or e > 20:
             continue
          else:
             conta += 1
    return conta
def obtem_cellsize(raster):
    from qgis.core import QgsRasterLayer
    rasterLyr = QgsRasterLayer(raster, "lyr")
    x = rasterLyr.rasterUnitsPerPixelX()
    return x
def nr_pixel_classe(raster):
    from osgeo import gdal
    import numpy
    img = gdal.Open(raster)
    array = numpy.array(img.ReadAsArray())
    values = numpy.unique(array)
    lista = []
    for i in values:
       if i < 0 or i > 20:
          continue
       else:
          lista.append(i)
    dic = {}
    array = img.ReadAsArray()
    for i in lista:
       conta = 0
       for e in array:
          temp = list(e)
          elementos = temp.count(i)
          conta += elementos
       dic_temp = {i:conta}
       dic.update(dic_temp)
```

```
    return dic
def indpCondi_all_QGIS(movimentos, workspace, saida):
    import math
    ExpNumTP = float(itera_shape(movimentos))
    factores = listar_raster(workspace)
    temp = []
    for i in factores:
       extensao = i[-4:]
       if extensao == '.tif':
          temp.append(i)
    factores = temp
    del temp
    nr_pixeis = float(conta_celula(workspace + "\\" + str(factores[0])))
    cellsize = float(obtem_cellsize(workspace + "\\" + str(factores[0])))
    unit_area=((((cellsize * cellsize) * float(nr_pixeis)) / 1000000.0) / ExpNumTP) /40.0
    ConvFac = (cellsize**2) / 1000000.0 / unit_area
    import xlwt
    excel = xlwt.Workbook()
    add_livro = excel.add_sheet('independencia')
    nr_linha = 1
    nr_linha_2 = len(factores) + 3
    nr_linha_3 = len(factores) + 3 + len(factores) +3
    for i in range(len(factores)):
       e = i + 1
       add_livro.write(0, e, factores[i])
       add_livro.write(e, 0, factores[i])
    for i in range(len(factores)):
       lista_overall = []
       lista_ric = []
       lista_tac = []
       class_count = nr_pixel_classe(workspace + "\\" + factores[i])
       PredT = 0.0
       for chave in class_count.keys():
          value = float(chave)
          count = class_count[chave]
          PredT += (value * count)
       del value, count, class_count
       PredT *= ConvFac
       for e in range(len(factores)):
          TVar = 0.0
          class_count = nr_pixel_classe(workspace + "\\" + factores[e])
          for chave in class_count.keys():
             value = float(chave)
             count = class_count[chave]
             TVar += (value * count * ConvFac)**2
          TStd = math.sqrt(TVar)
          del value, count, class_count
          TS = (PredT - ExpNumTP) / TStd
          n = ExpNumTP
          T = PredT
          P = acessoria(TS) * 100.0
          if P > 50.0:
             overallCI = 100.0 * (100.0 - P) / 50.0
          else:
             overallCI = 100.0 * (100.0 - (50.0 + (50.0 - P))) / 50.0
          lista_overall.append(overallCI)
          ric = n/T
          lista_ric.append(ric)
          lista_tac.append(P)
       for i in range(len(lista_overall)):
          e = i + 1
          add_livro.write(nr_linha, e, lista_overall[i])
          add_livro.write(nr_linha_2, e, lista_ric[i])
          add_livro.write(nr_linha_3, e, lista_tac[i])
       nr_linha += 1
```

```
        nr_linha_2 += 1
        nr_linha_3 += 1
    excel.save(saida)
indpCondi_all_QGIS(fenomeno, _area_trabalho, folha)
```

Programa 22 - Bloco de código (em Python, v. 2.7.3) que executa o cálculo dos indicadores de (in)dependência condicional entre os vários factores condicionantes – GDAL/OGR 1.11.1.

```
""" Calcula a independência entre as várias variáveis que compõem um modelo que tem por
objectivo avaliar a susceptibilidade de um território a movimentos de vertente - devolve
uma matriz em folha de cálculo com os resultados - GDAL"""
import time
from osgeo import gdal
from osgeo import ogr
import math, xlwt
def acessoria(Z):
      import math
      pi = 3.141592654
      const6 = 0.2316419
      b1 = 0.31938153
      b2 = -0.356563782
      b3 = 1.781477937
      b4 = -1.821255978
      b5 = 1.330274429
      Z = float(Z)
      X = Z
      if X < 0.0:
         X = -Z
      t = 1.0 / (const6 * X + 1.0)
      pid = 2.0 * pi
      XX = -X * X / 2.0
      XX = math.exp(XX) / math.sqrt(pid)
      PZ = (b1 * t) + (b2 * t *t) + (b3 * (t**3)) + (b4 * (t**4)) + (b5 * (t**5))
      PZ = 1.0 - (PZ * XX)
      if Z < 0:
         PZ = 1.0 - PZ
      return PZ
def itera_shape(shapefile):
      driver = ogr.GetDriverByName("ESRI Shapefile")
      dataSource = driver.Open(shapefile, 0)
      layer = dataSource.GetLayer()
      conta = 0
      for feature in layer:
         conta += 1
      return conta
def listar_raster(path):
      from os import walk
      r = []
      for (dirpath, dirnames, filenames) in walk(path):
         r.extend(filenames)
         break
      return r
def conta_nrpixeis(raster):
      img = gdal.Open(raster)
      array = img.ReadAsArray()
      conta = 0
      for i in array:
         for e in i:
            if e < 0 or e > 20:
               continue
            else:
```

```
            conta += 1
    return conta
def obtem_cellsize(raster):
    img = gdal.Open(raster)
    (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) =
    img.GetGeoTransform()
    return x_size
def nr_pixel_classe(raster):
    import numpy
    img = gdal.Open(raster)
    array = numpy.array(img.ReadAsArray())
    values = numpy.unique(array)
    lista = []
    for i in values:
       if i < 0 or i > 20:
          continue
       else:
          lista.append(i)
    dic = {}
    array = img.ReadAsArray()
    for i in lista:
       conta = 0
       for e in array:
          temp = list(e)
          elementos = temp.count(i)
          conta += elementos
          dic_temp = {i:conta}
          dic.update(dic_temp)
    return dic
def indpCondi_all_GDAL(movimentos, workspace, saida):
    ExpNumTP = float(itera_shape(movimentos))
    factores = listar_raster(workspace)
    temp = []
    for i in factores:
       extensao = i[-4:]
          if extensao == '.tif':
             temp.append(i)
    factores = temp
    del temp
    nr_pixeis = float(conta_nrpixeis(str(workspace) + '\\' + str(factores[0])))
    cellsize = obtem_cellsize(str(workspace) + '\\' + str(factores[0]))
    area_pixel = cellsize * cellsize
    area_total = area_pixel * nr_pixeis
    area_kilometros = area_total / 1000000.0
    unit_area = (area_kilometros / ExpNumTP) / 40.0
    unit_area = float(unit_area)
    ConvFac = (cellsize**2) / 1000000.0 / unit_area
    excel = xlwt.Workbook()
    add_livro = excel.add_sheet('independencia')
    nr_linha = 1
    nr_linha_2 = len(factores) + 3
    nr_linha_3 = len(factores) + 3 + len(factores) +3
    for i in range(len(factores)):
       e = i + 1
       add_livro.write(0, e, factores[i])
       add_livro.write(e, 0, factores[i])
    for i in range(len(factores)):
       lista_overall = []
       lista_ric = []
       lista_tac = []
       class_count = nr_pixel_classe(workspace + "\\" + factores[i])
       PredT = 0.0
       for chave in class_count.keys():
          value = float(chave)
```

```
            count = class_count[chave]
            PredT += (value * count)
         del value, count, class_count
         PredT *= ConvFac
         for e in range(len(factores)):
            TVar = 0.0
            class_count = nr_pixel_classe(workspace + "\\" + factores[e])
            for chave in class_count.keys():
               value = float(chave)
               count = class_count[chave]
               TVar += (value * count * ConvFac)**2
            TStd = math.sqrt(TVar)
            del value, count, class_count
            TS = (PredT - ExpNumTP) / TStd
            n = ExpNumTP
            T = PredT
            P = acessoria(TS) * 100.0
            if P > 50.0:
               overallCI = 100.0 * (100.0 - P) / 50.0
            else:
               overallCI = 100.0 * (100.0 - (50.0 + (50.0 - P))) / 50.0
            lista_overall.append(overallCI)
            ric = n/T
            lista_ric.append(ric)
            lista_tac.append(P)
         for i in range(len(lista_overall)):
            e = i + 1
            add_livro.write(nr_linha, e, lista_overall[i])
            add_livro.write(nr_linha_3, e, lista_tac[i])
            add_livro.write(nr_linha_2, e, lista_ric[i])
         nr_linha += 1
         nr_linha_2 += 1
         nr_linha_3 += 1
      excel.save(saida)
```

Programa 23 - Algoritmo (em Python, v. 2.7.3) para a criação, para cada factor condicionante, de amostras de treino (70%) e amostras de validação (30%) que respeitam esta repartição pelas várias variáveis – ArcGIS 10.2 e ArcPy.

```
# Criação de amostras de treino (70%) e amostras de avaliação (30%) consideradas ideais
para cada um dos factores condicionantes - por ideal entenda-se: cada classe de
cada factor tem 70% dos pontos que a intersectam definidos como ocorrências para
treino e tem 30% para validação – ArcGIS
import arcpy, os, unicodedata, shutil, xlwt
movimentos = arcpy.GetParameterAsText(0)
factores = arcpy.GetParameterAsText(1)
prj = arcpy.GetParameterAsText(2)
relatorio = arcpy.GetParameterAsText(3)
arcpy.env.overwriteOutput = True
def listar_factores(work_factores):
      arcpy.env.workspace = work_factores
      list_fac = arcpy.ListRasters()
      list_unicode = []
      for i in list_fac:
         e = unicodedata.normalize('NFKD', i).encode('ascii', 'ignore')
         o = work_factores + "\\" + e
         list_unicode.append(o)
      return list_unicode
def cria_layer(objecto, nome_layer, rst_ou_vect):
      if rst_ou_vect == True:
         lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
```

```
        return lyr
     else:
        lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
        return lyr
def rst2Polygon(entrada, saida):
     lyr = cria_layer(entrada, "rst_lyr", True)
     arcpy.RasterToPolygon_conversion(lyr, saida, "NO_SIMPLIFY", "Value")
     del lyr
def dissolve(entrada, saida):
     lyr = cria_layer(entrada, "lyr", False)
     arcpy.Dissolve_management(lyr, saida, "GRIDCODE", "", "MULTI_PART", "")
     del lyr
def cria_amostras_classe(factor, fenomeno, geral, fsetenta, ftrinta, ref_esp):
    arcpy.env.workspace = geral
    layer = cria_layer(factor, "rst_factor", False)
    cursor = arcpy.SearchCursor(layer)
    linha = cursor.next()
    while linha:
       classe = int(linha.getValue("GRIDCODE"))
       selected = "classe_" + str(classe) + ".shp"
       arcpy.Select_analysis(layer, selected, "GRIDCODE = " + str(classe))
       intersected = "nr_pontos_" + str(classe) + ".shp"
       arcpy.Intersect_analysis([selected, fenomeno], intersected, "", "", "")
       nr_pontos = 0
       cs = arcpy.SearchCursor(intersected)
       ls = cs.next()
       while ls:
          nr_pontos += 1
          ls = cs.next()
       del cs, ls
       if nr_pontos < 4:
          linha = cursor.next()
          continue
       setenta = int((nr_pontos * 70) / 100)
       cs = arcpy.SearchCursor(intersected)
       ls = cs.next()
       nr_id = 0
       nr_id_list = []
       pntX = []
       pntY = []
       for i in range(setenta):
          ponto = ls.Shape.centroid
          nr_id_list.append(nr_id)
          pntX.append(ponto.X)
          pntY.append(ponto.Y)
          nr_id += 1
          ls = cs.next()
       del cs, ls, nr_id, ponto
       arcpy.CreateFeatureclass_management(fsetenta, "setenta_" + str(classe) + ".shp",
       "POINT", "", "", "ENABLED", ref_esp)
       cs = arcpy.InsertCursor(fsetenta + "\\setenta_" + str(classe) + ".shp")
       ponto = arcpy.CreateObject("Point")
       for i in range(len(nr_id_list)):
          ponto.ID = nr_id_list[i]
          ponto.X = pntX[i]
          ponto.Y = pntY[i]
          inserir = cs.newRow()
          inserir.Shape = ponto
          cs.insertRow(inserir)
       del cs, ponto, inserir, pntX, pntY, nr_id_list
       arcpy.Erase_analysis(intersected, fsetenta + "\\setenta_" + str(classe) + ".shp",
       ftrinta + "\\trinta_" + str(classe) + ".shp")
       linha = cursor.next()
def cria_amostras_factor(fsetenta, ftrinta, conta, dsetenta, dtrinta):
```

```
    arcpy.env.workspace = fsetenta
    lst_setenta = arcpy.ListFeatureClasses()
    arcpy.env.workspace = ftrinta
    lst_trinta = arcpy.ListFeatureClasses()
    temp_1 = []
    temp_2 = []
    for i in range(len(lst_setenta)):
       e = unicodedata.normalize('NFKD', lst_setenta[i]).encode('ascii', 'ignore')
       o = fsetenta + "\\" + e
       temp_1.append(o)
       h = unicodedata.normalize('NFKD', lst_trinta[i]).encode('ascii', 'ignore') j =
       ftrinta + "\\" + h
       temp_2.append(j)
    lst_setenta = temp_1
    lst_trinta = temp_2
    del temp_1, temp_2, i, e, o, h, j
    for i in range(len(lst_setenta)):
       if i == 0:
          string_70 = lst_setenta[i]
          string_30 = lst_trinta[i]
       else:
          string_70 = string_70 + ";" + lst_setenta[i]
          string_30 = string_30 + ";" + lst_trinta[i]
    arcpy.Merge_management(string_70, dsetenta + "\\factor70_" + str(conta) + ".shp")
    arcpy.Merge_management(string_30, dtrinta + "\\factor30_" + str(conta) + ".shp")
def main_job_ArcGIS(fenomeno, workspace_factores, ref_espacial, saida):
    lst_factores = listar_factores(workspace_factores)
    os.mkdir("C:\\rst2shp")
    os.mkdir("C:\\diss")
    conta = 0
    t = []
    for i in lst_factores:
       rst2Polygon(i, "C:\\rst2shp\\shp_factor_" + str(conta) + ".shp")
       dissolve("C:\\rst2shp\\shp_factor_" + str(conta) + ".shp", "C:\\diss\\d_factor_" +
       str(conta) + ".shp")
       t.append("C:\\diss\\d_factor_" + str(conta) + ".shp")
       conta += 1
    lst_factores = t
    del conta, t
    areatrab = "C:\\areatrab"
    os.mkdir(areatrab)
    caminho_setenta_factor = areatrab + "\\factor_setenta"
    caminho_trinta_factor = areatrab + "\\factor_trinta"
    os.mkdir(caminho_setenta_factor)
    os.mkdir(caminho_trinta_factor)
    logos = 0
    lyr_fenomeno = cria_layer(fenomeno, "fenomeno", False)
    for i in lst_factores:
       caminho_setenta_classe = areatrab + "\\classe_70"
       caminho_trinta_classe = areatrab + "\\classe_30"
       os.mkdir(caminho_setenta_classe)
       os.mkdir(caminho_trinta_classe)
       cria_amostras_classe(i, lyr_fenomeno, areatrab, caminho_setenta_classe,
       caminho_trinta_classe, ref_espacial)
       cria_amostras_factor(caminho_setenta_classe, caminho_trinta_classe, logos,
       caminho_setenta_factor, caminho_trinta_factor)
       shutil.rmtree(caminho_setenta_classe)
       shutil.rmtree(caminho_trinta_classe)
       logos += 1
       del logos
    excel = xlwt.Workbook()
    arcpy.env.workspace = areatrab
    for amostra in range(len(lst_factores)):
       nr_linha = 0
```

```
        add_livro = excel.add_sheet("amostra_" + str(amostra))
        amostra_70 = cria_layer(caminho_setenta_factor + "\\factor70_" + str(amostra) +
        ".shp", "amostragem70", False)
        amostra_30 = cria_layer(caminho_trinta_factor + "\\factor30_" + str(amostra) +
        ".shp", "amostragem30", False)
        for raster in lst_factores:
            add_livro.write(nr_linha, 0, raster)
            nr_linha += 1
            layer = cria_layer(raster, "factor", False)
            cursor = arcpy.SearchCursor(layer)
            linha = cursor.next()
            add_livro.write(nr_linha, 1, 'ponto_70')
            add_livro.write(nr_linha, 2, 'ponto_30')
            nr_linha += 1
            while linha:
                classe = int(linha.getValue("GRIDCODE"))
                feature_layer = "classe_" + str(classe) + ".shp"
                arcpy.Select_analysis(layer, feature_layer, "GRIDCODE = " + str(classe))
                intersect_70 = "nr_pontos_" + str(classe) + ".shp"
                arcpy.Intersect_analysis([feature_layer, amostra_70], intersect_70)
                cs = arcpy.SearchCursor(cria_layer(intersect_70, "intersect_70", False))
                ls = cs.next()
                conta_70 = 0
                while ls:
                    conta_70 += 1
                    ls = cs.next()
                del cs, ls
                intersect_30 = "nr_pontos30_" + str(classe) + ".shp"
                arcpy.Intersect_analysis([feature_layer, amostra_30], intersect_30)
                cs = arcpy.SearchCursor(cria_layer(intersect_30, "intersect_30", False))
                ls = cs.next()
                conta_30 = 0
                while ls:
                    conta_30 += 1
                    ls = cs.next()
                del cs, ls
                add_livro.write(nr_linha, 0, classe)
                add_livro.write(nr_linha, 1, conta_70)
                add_livro.write(nr_linha, 2, conta_30)
                nr_linha += 1
                linha = cursor.next()
    excel.save(saida)
main_job_ArcGIS(movimentos, factores, prj, relatorio)
```

Programa 24 - Algoritmo (em Python, v. 2.7.4) para a criação, para cada factor condicionante, de amostras de treino (70%) e de validação (30%) que respeitam esta repartição pelas várias variáveis – GRASS GIS 7.0.0 e PyGRASS.

```
# Criação de amostras de treino (70%) e amostras de avaliação (30%) consideradas
ideais para cada um dos factores condicionanes - por ideal entenda-se: cada
classe de cada factor tem 70% dos pontos que a intersectam definidos como
ocorrências para treino e tem 30% para validação - GRASS GIS
def r_in_gdal(raster, grass_raster):
    from grass.pygrass.modules import Module
    adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
    overwrite=True, run_=False)
    adiciona()
def lista_factores(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
```

```
            r.extend(filenames)
            break
        t = []
        for i in r:
            ext = i[-4:]
            if ext == '.tif':
                t.append(i)
        f = []
        for i in t:
            r_in_gdal(path + "\\" + i, i[:-4])
            f.append(i[:-4])
        return f
def RasterToPolygon(raster, vector):
        from grass.pygrass.modules import Module
        raster2poly = Module("r.to.vect", input=raster, output=vector, type='area',
        column='value', overwrite=True, run_=False)
        raster2poly()
def cria_objTable(tab, path):
        from grass.pygrass.vector.table import Table, get_path
        import sqlite3
        objTabela = Table(name=tab, connection=sqlite3.connect(get_path(path)))
        return objTabela
def id_classes(in_vector, field, grassdata_workspace, location, mapset):
        caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
        "\\sqlite\\sqlite.db"
        tabela = cria_objTable(in_vector, caminho)
        cls = []
        for linha in tabela:
            cls.append(int(linha[1]))
        import numpy
        cls = numpy.unique(cls)
        t = []
        for i in cls:
            if i < 0 or i > 20:
                continue
            else:
                t.append(i)
        return t
def extract(vector, name_output, classe):
        from grass.pygrass.modules import Module
        onde = "value = " + str(classe)
        extrai = Module("v.extract", input=vector, type="area", where=onde, output=
        name_output, overwrite=True, run_=False)
        extrai()
def v_in_ogr(shapefile, grass_vector):
        from grass.pygrass.modules import Module
        adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
        overwrite=True, run_=False)
        adiciona()
def intersection(pontos, poligonos, saida):
        from grass.pygrass.modules import Module
        intersect = Module("v.select", ainput=pontos, atype='point', binput=poligonos,
        btype='area', output=saida, operator='intersects', overwrite=True, run_=False)
        intersect()
def conta_numero_pnt(entrada):
        from grass.pygrass.vector import VectorTopo
        layer = VectorTopo(entrada)
        layer.is_open()
        layer.exist()
        layer.mapset
        layer.open(mode='r')
        try:
            linha = layer.next()
```

```
    except:
       return
    conta = 0
    try:
       while linha:
          conta += 1
          linha = layer.next()
    except:
       layer.close()
    return conta
def extrai_70_30(vector, setenta):
    lst70 = []
    lst30 = []
    from grass.pygrass.vector import VectorTopo
    layer = VectorTopo(vector)
    layer.open(mode='r')
    linha = layer.next()
    logos = 1
    try:
       while linha:
          ponto = layer.read(logos)
          string = str.split(str(ponto), "(")
          outra = str.split(string[1], " ")
          x = float(outra[0])
          y = float(outra[1][:-1])
          if logos <= setenta:
             lst70.append(temp)
          elif logos > setenta:
             lst30.append(temp)
          linha = layer.next()
          logos += 1
    except:
       layer.close()
    return lst70, lst30
def cria_vector_pnt(lista, saida):
    from grass.pygrass.vector import VectorTopo
    from grass.pygrass.vector.geometry import Point
    new = VectorTopo(saida)
    cols = [(u'cat', 'INTEGER PRIMARY KEY')]
    new.open('w', tab_name=saida, tab_cols=cols)
    for pnt in lista:
       point = Point(pnt[0], pnt[1])
       new.write(point)
    new.table.conn.commit()
    new.table.execute().fetchall()
    new.close()
def merge_pntLayers(lista, saida):
    pontos = []
    from grass.pygrass.vector import VectorTopo
    for i in lista:
       layer = VectorTopo(i)
       layer.open(mode='r')
       linha = layer.next()
       logos = 1
       try:
          while linha:
             pnt = layer.read(logos)
             string = str.split(str(pnt), "(")
             outra = str.split(string[1], " ")
             temp = [float(outra[0]), float(outra[1][:-1])]
             pontos.append(temp)
             linha = layer.next()
             logos += 1
       except:
```

```
            layer.close()
    cria_vector_pnt(pontos, saida)
def gera_amostras_main(fenomeno, workspace_factores, saida, grassdata, location,
mapset):
    lst_factores = lista_factores(workspace_factores)
    conta = 0
    v_in_ogr(fenomeno, "movimentos")
    lst_de_lst = []
    for i in lst_factores:
       RasterToPolygon(i, "v_factor_" + str(conta))
       lst_cls = id_classes("v_factor_" + str(conta), "value", grassdata, location,
       mapset)
       lst_extract = []
       for cls in lst_cls:
          extract("v_factor_" + str(conta), "factor_" + str(conta) + "_" + str(cls), cls)
          lst_extract.append("factor_" + str(conta) + "_" + str(cls))
       lst_de_lst.append(lst_extract)
       lst_intersect = []
       for i in range(len(lst_extract)):
          intersection("movimentos", lst_extract[i], "intersect_" + str(conta) + "_" +
          str(i))
          lst_intersect.append("intersect_" + str(conta) + "_" + str(i))
       c = 0
       lst_amostras_classe70 = []
       lst_amostras_classe30 = []
       for i in lst_intersect:
          nr_pontos = conta_numero_pnt(i)
          if nr_pontos < 4:
             continue
          setenta = int((nr_pontos * 70) / 100)
          amostras = extrai_70_30(i, setenta)
          cria_vector_pnt(amostras[0], "setenta_" + str(conta) + "_" + str(c))
          lst_amostras_classe70.append("setenta_" + str(conta) + "_" + str(c))
          cria_vector_pnt(amostras[1], "trinta_" + str(conta) + "_" + str(c))
          lst_amostras_classe30.append("trinta_" + str(conta) + "_" + str(c))
          c += 1
       merge_pntLayers(lst_amostras_classe70, "amostra70_" + str(conta))
       merge_pntLayers(lst_amostras_classe30, "amostra30_" + str(conta))
       conta += 1
    import xlwt
    excel = xlwt.Workbook()
    for amostra in range(len(lst_factores)):
       nr_linha = 0
       add_livro = excel.add_sheet("amostra_" + str(amostra))
       amostra_70 = "amostra70_" + str(amostra)
       amostra_30 = "amostra30_" + str(amostra)
       for raster in range(len(lst_factores)):
          add_livro.write(nr_linha, 0, lst_factores[raster])
          nr_linha += 1
          add_livro.write(nr_linha, 1, 'ponto_70')
          add_livro.write(nr_linha, 2, 'ponto_30')
          conta = 0
          nr_linha += 1
          lst_shp = lst_de_lst[raster]
          for classe in range(len(lst_shp)):
             intersection(amostra_70, lst_shp[classe], "intersect70_" + str(amostra) + "_" +
             str(raster) + "_" + str(conta))
             conta_70 = conta_numero_pnt("intersect70_" + str(amostra) + "_" + str(raster) +
             "_" + str(conta))
             intersection(amostra_30, lst_shp[classe], "intersect30_" + str(amostra) + "_" +
             str(raster) + "_" + str(conta))
             conta_30 = conta_numero_pnt("intersect30_" + str(amostra) + "_" + str(raster) +
             "_" + str(conta))
```

```
            add_livro.write(nr_linha, 0, classe)
            add_livro.write(nr_linha, 1, conta_70)
            add_livro.write(nr_linha, 2, conta_30)
            nr_linha += 1
            conta += 1
    excel.save(saida)
```

Programa 25 - Algoritmo (em Python, v. 2.7.5) para a criação, para cada factor condicionante, de amostras de treino (70%) e amostras de validação (30%) que respeitam esta repartição pelas várias variáveis – QGIS 2.10.1, qgis.core e processing.

```
# Criação de amostras de treino (70%) e amostras de avaliação (30%) consideradas
ideais para cada um dos factores condicionanes - por ideal entenda-se: cada
classe de cada factor tem 70% dos pontos que a intersectam definidos como
ocorrências para treino e tem 30% para validação – Qgis
import os, shutil, xlwt
from qgis.core import QgsPoint, QgsGeometry, QgsFeature, QgsProviderRegistry
from osgeo import gdal, ogr
##MovVert= group
##AmostraIdeal= name
##movimentos= vector
##factores= folder
##relatorio= output file
def lista_factores(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
       r.extend(filenames)
       break
    t = []
    for i in r:
       ext = i[-4:]
       if ext == '.tif':
          t.append(path + "\\" + i)
    return t
def cria_layer(objecto, nome, rst_ou_vect):
    if rst_ou_vect == True:
       from qgis.core import QgsRasterLayer
       rst_lyr = QgsRasterLayer(objecto, nome)
       return rst_lyr
    elif rst_ou_vect == False:
       from qgis.core import QgsVectorLayer
       v_lyr = QgsVectorLayer(objecto, nome, "ogr")
       return v_lyr
def rst2polygon(entrada, saida):
    import processing
    processing.runalg("gdalogr:polygonize", entrada, "value", saida)
def criar_shp_aPartirDeOutra(entrada, destino, momento):
    from qgis.core import QgsVectorFileWriter
    from osgeo import ogr
    lyr = cria_layer(entrada, "indf", False)
    provider = lyr.dataProvider()
    if momento == 1:
       writer = QgsVectorFileWriter(destino, provider.encoding(), provider.fields(),
       ogr.wkbPolygon,provider.crs())
    elif momento == 2:
       writer = QgsVectorFileWriter(destino, provider.encoding(), provider.fields(),
       ogr.wkbPoint,provider.crs())
def split(entrada, workspace_saida):
    import processing
```

```
    processing.runalg("qgis:splitvectorlayer", entrada, "value", workspace_saida)
def lista_shapefile(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
        r.extend(filenames)
        break
    t = []
    for i in r:
       ext = i[-4:]
       if ext == '.shp':
          t.append(path + "\\" + i)
    return t
def intersection(pontos, poligonos, saida):
    import processing
    processing.runalg("qgis:intersection", pontos, poligonos, saida)
def cria_prj(f_prj, saida):
    texto = open(f_prj, 'r')
    for linha in texto.readlines():
       coord = linha
    texto.close() new_prj = open(saida, "w")
    new_prj.write(str(coord))
    new_prj.close()
def cria_amostras_classe(factor, fenomeno, geral, fsetenta, ftrinta, ref_esp,
contador):
    from qgis.core import QgsPoint, QgsProviderRegistry, QgsGeometry
    temp = 'C:\\temp' + str(contador)
    os.mkdir(temp)
    split(factor, 'C:\\rst2shp' + str(contador))
    lst_shp = lista_shapefile(' C:\\rst2shp' + str(contador))
    conta = 0
    for classe in range(1, len(lst_shp)):
       intersection(fenomeno, lst_shp[classe], temp + "\\intersect" + str(conta) +
       ".shp")
       cria_prj(ref_esp, temp + "\\intersect" + str(conta) + ".prj")
       nr_pontos = 0
       registo = QgsProviderRegistry.instance()
       provider = registo.provider("ogr", temp + "\\intersect" + str(conta) + ".shp")
       for feature in provider.getFeatures():
          nr_pontos += 1
       del registo, provider
       if nr_pontos < 4:
          conta += 1
          continue
       setenta = int((nr_pontos * 70) / 100)
       dic70 = []
       dic30 = []
       registo = QgsProviderRegistry.instance()
       provider = registo.provider("ogr", temp + "\\intersect" + str(conta) + ".shp")
       logos = 1
       for feature in provider.getFeatures():
          geom = feature.geometry()
          ponto = list(geom.asPoint())
          dtemp = [ponto[0], ponto[1]]
          if logos <= setenta:
             dic70.append(dtemp)
          elif logos > setenta:
             dic30.append(dtemp)
          logos += 1
       del dtemp
       criar_shp_aPartirDeOutra(temp + "\\intersect" + str(conta) + ".shp", fsetenta +
       "\\setenta_" + str(conta) + ".shp", 2)
       lyr70 = cria_layer(fsetenta + "\\setenta_" + str(conta) + ".shp", "lyr70", False)
       vpr = lyr70.dataProvider()
```

```
        for coord in dic70:
            pnt = QgsGeometry.fromPoint(QgsPoint(coord[0], coord[1]))
            f = QgsFeature()
            f.setGeometry(pnt)
            vpr.addFeatures([f])
            lyr70.updateExtents()
        criar_shp_aPartirDeOutra(temp + "\\intersect" + str(conta) + ".shp",ftrinta +
        "\\trinta_" + str(conta) + ".shp", 2)
        lyr30 = cria_layer(ftrinta + "\\trinta_" + str(conta) + ".shp", "lyr30",False)
        vpr = lyr30.dataProvider()
        for coord in dic30:
            pnt = QgsGeometry.fromPoint(QgsPoint(coord[0], coord[1]))
            f = QgsFeature()
            f.setGeometry(pnt)
            vpr.addFeatures([f])
            lyr30.updateExtents()
        conta += 1
def cria_amostras_factor(fsetenta, ftrinta, conta, dsetenta, dtrinta, ref_esp):
    import processing
    files70 = lista_shapefile(fsetenta)
    files30 = lista_shapefile(ftrinta)
    for i in range(len(files70)):
        if i == 0:
            string70 = files70[i]
            string30 = files30[i]
        else:
            string70 = string70 + ";" + files70[i]
            string30 = string30 + ";" + files30[i]
    out70 = dsetenta + "\\factor70_" + str(conta) + ".shp"
    out30 = dtrinta + "\\factor30_" + str(conta) + ".shp"
    processing.runalg("saga:mergelayers", string70, True, True, out70)
    cria_prj(ref_esp, dsetenta + "\\factor70_" + str(conta) + ".prj")
    processing.runalg("saga:mergelayers", string30, True, True, out30)
    cria_prj(ref_esp, dtrinta + "\\factor30_" + str(conta) + ".prj")
def main_job_Qgis(fenomeno, workspace_factores, saida):
    prj = fenomeno[:-4] + ".prj"
    from qgis.core import QgsProviderRegistry
    lst_factores = lista_factores(workspace_factores)
    conta = 0
    t = []
    for i in lst_factores:
        os.mkdir("C:\\rst2shp" + str(conta))
        rst2polygon(i, "C:\\rst2shp" + str(conta) + "\\factor_" + str(conta) + ".shp")
        cria_prj(prj, C:\\rst2shp" + str(conta) + "\\factor_" + str(conta) + ".prj")
        t.append("C:\\rst2shp" + str(conta) + "\\factor_" + str(conta) + ".shp")
        conta += 1
    lst_factores = t
    del t, conta
    areatrab = "C:\\areatrab"
    os.mkdir(areatrab)
    caminho_setenta_factor = areatrab + "\\factor_setenta"
    caminho_trinta_factor = areatrab + "\\factor_trinta"
    os.mkdir(caminho_setenta_factor)
    os.mkdir(caminho_trinta_factor)
    logos = 0
    lyr_fenomeno = cria_layer(fenomeno, "fenomeno", False)
    for i in lst_factores:
        caminho_setenta_classe = areatrab + "\\classe_70" + str(logos)
        caminho_trinta_classe = areatrab + "\\classe_30" + str(logos)
        os.mkdir(caminho_setenta_classe)
        os.mkdir(caminho_trinta_classe)
        cria_amostras_classe(i, lyr_fenomeno, areatrab, caminho_setenta_classe,
        caminho_trinta_classe, prj, logos)
        cria_amostras_factor(caminho_setenta_classe, caminho_trinta_classe, logos,
        caminho_setenta_factor, caminho_trinta_factor, prj)
```

```
        logos += 1
    del logos
    excel = xlwt.Workbook()
    for amostra in range(len(lst_factores)):
       nr_linha = 0
       add_livro = excel.add_sheet("amostra_" + str(amostra))
       amostra_70 = cria_layer(caminho_setenta_factor + "\\factor70_" + str(amostra) +
       ".shp", "amostragem70", False)
       amostra_30 = cria_layer(caminho_trinta_factor + "\\factor30_" + str(amostra) +
       ".shp", "amostragem30", False)
       for raster in range(len(lst_factores)):
          pasta = areatrab + "\\temp" + str(amostra) + "_" + str(raster)
          os.mkdir(pasta)
          add_livro.write(nr_linha, 0, lst_factores[raster])
          nr_linha += 1
          lst_shp = lista_shapefile("C:\\dfactor" + str(raster))
          add_livro.write(nr_linha, 1, 'ponto_70')
          add_livro.write(nr_linha, 2, 'ponto_30')
          conta = 0
          nr_linha += 1
          for classe in range(1, len(lst_shp)):
             intersection(amostra_70, lst_shp[classe], pasta + "\\intersect70_" + str(conta)
             + ".shp")
             registo = QgsProviderRegistry.instance()
             provider = registo.provider("ogr", pasta + "\\intersect70_" + str (conta) +
             ".shp")
             conta_70 = 0
             for feature in provider.getFeatures():
                conta_70 += 1
             intersection(amostra_30, lst_shp[classe], pasta + "\\intersect30_" + str(conta)
             + ".shp")
             conta_30 = 0
             registo = QgsProviderRegistry.instance()
             provider = registo.provider("ogr", pasta + "\\intersect30_" + str(conta) +
             ".shp")
             for feature in provider.getFeatures():
                conta_30 += 1
             add_livro.write(nr_linha, 0, classe)
             add_livro.write(nr_linha, 1, conta_70)
             add_livro.write(nr_linha, 2, conta_30)
             conta += 1
             nr_linha += 1
    excel.save(saida)
main_job_Qgis(movimentos, factores, relatorio)
```

Programa 26 - Algoritmo (em Python, v. 2.7.3) para a criação, para cada factor condicionante, de amostras de treino (70%) e amostras de validação (30%) que respeitam esta repartição pelas várias variáveis – GDAL 1.11.1.

```
# Criação de amostras de treino (70%) e amostras de avaliação (30%) consideradas
ideais para cada um dos factores condicionanes - por ideal entenda-se: cada
classe de cada factor tem 70% dos pontos que a intersectam definidos como
ocorrências para treino e tem 30% para validação – GDAL
from osgeo import gdal, ogr
import os, shutil, xlwt, numpy
def lista_factores(path):
     from os import walk
     r = []
     for (dirpath, dirnames, filenames) in walk(path):
        r.extend(filenames)
        break
     t = []
     for i in r:
```

```
        ext = i[-4:]
        if ext == '.tif':
           t.append(path + "\\" + i)
     return t
def lista_shapefile(path):
     from os import walk
     r = []
     for (dirpath, dirnames, filenames) in walk(path):
        r.extend(filenames)
        break
     t = []
     for i in r:
        ext = i[-4:]
        if ext == '.shp':
           t.append(path + "\\" + i)
     return t
def rst2Polygon(entrada, saida):
     img = gdal.Open(entrada)
     band = img.GetRasterBand(1)
     drv = ogr.GetDriverByName("ESRI SHAPEFILE")
     shp = drv.CreateDataSource(saida)
     shp_lyr = shp.CreateLayer(saida, srs=None)
     shp_lyr.CreateField(ogr.FieldDefn("value", ogr.OFTInteger))
     gdal.Polygonize(band, None, shp_lyr, 0, [], callback=None)
def cria_prj(f_prj, saida):
     texto = open(f_prj, 'r')
     for linha in texto.readlines():
        coord = linha
     texto.close()
     new_prj = open(saida, "w")
     new_prj.write(str(coord))
     new_prj.close()
def id_classe(in_shp, field):
     drv = ogr.GetDriverByName("ESRI SHAPEFILE")
     shp = drv.Open(in_shp, 0)
     lyr = shp.GetLayer()
     c = []
     for feature in lyr:
        classe = feature.GetField(field)
        c.append(int(classe))
     classes = numpy.unique(c)
     return classes
def extrai(entrada, variavel, campo, logos):
     drv = ogr.GetDriverByName("ESRI SHAPEFILE")
     shp = drv.Open(entrada, 0)
     lyr = shp.GetLayer()
     i_poligonos = {}
     i_ilhas = {}
     id_feature = 0
     for feature in lyr:
        identificacao = feature.GetField(campo)
        if int(identificacao) == variavel:
           poligono = 1
           geom = feature.GetGeometryRef()
           lpolygon = []
           ilhas = []
           for polygon in geom:
              if poligono == 1:
                 figura = str(polygon)
                 lista = str.split(figura, ",")
                 c = 0
                 for string in lista:
                    if c == 0:
                       temp = str.split(string, "(")
```

```
                        par = temp[1]
                     if c >= 1 and c < len(lista) -1:
                        par = string
                     if c == len(lista) -1:
                        temp = str.split(string, ")")
                        par = temp[0]
                     lpolygon.append(par)
                     c += 1
               else:
                  ilha = []
                  figura = str(polygon)
                  lista = str.split(figura, ",")
                  c = 0
                  for string in lista:
                     if c == 0:
                        temp = str.split(string, "(")
                        par = temp[1]
                     elif c >= 1 and c < len(lista) -1:
                        par = string
                     elif c == len(lista) -1:
                        temp = str.split(string, ")")
                        par = temp[0]
                     c += 1
                     ilha.append(par)
                  ilhas.append(ilha)
               poligono += 1
            i_poligonos.update({id_feature:lpolygon})
            i_ilhas.update({id_feature:ilhas})
            id_feature += 1
      if logos == 1:
         return i_poligonos
      elif logos == 2:
         return i_ilhas
def criaSHPholes(dic_poligonos, dic_ilhas, saida):
   shpDriver = ogr.GetDriverByName("ESRI Shapefile")
   outDataSource = shpDriver.CreateDataSource(saida)
   outLayer= outDataSource.CreateLayer(saida, geom_type=ogr.wkbPolygon)
   featureDefn = outLayer.GetLayerDefn()
   for chave in dic_poligonos.keys():
      outFeature = ogr.Feature(featureDefn)
      out_ring = ogr.Geometry(ogr.wkbLinearRing)
      for coord in dic_poligonos[chave]:
         coord = str.split(coord, " ")
         x = float(coord[0])
         y = float(coord[1])
         out_ring.AddPoint(x, y)
      holes = []
      for ilha in dic_ilhas[chave]:
         inner_ring = ogr.Geometry(ogr.wkbLinearRing)
         for coord in ilha:
            coord = str.split(coord, " ")
            x = float(coord[0])
            y = float(coord[1])
            inner_ring.AddPoint(x, y)
         holes.append(inner_ring)
      poly = ogr.Geometry(ogr.wkbPolygon)
      poly.AddGeometry(out_ring)
      for i in holes:
         poly.AddGeometry(i)
      outFeature.SetGeometry(poly)
      outLayer.CreateFeature(outFeature)
def gdal_dissolve_split(inshp, field, contador):
   os.mkdir("C:\\factor_" + str(contador))
   lst_classes = id_classe(inshp, field)
```

```
    t = []
    for i in lst_classes:
       if i < 0 or i > 20:
          continue
       else:
          t.append(i)
    lst_classes = t
    del t
    numeracao = 0
    for classe in lst_classes:
       poligonos = extrai(inshp, classe, field, 1)
       buracos = extrai(inshp, classe, field, 2)
       criaSHPholes(poligonos, buracos, "C:\\factor_" + str(contador) + "\\classe_" +
       str(numeracao) + ".shp")
       numeracao += 1
def intersection(point, polygon, out):
    drv = ogr.GetDriverByName("ESRI SHAPEFILE")
    point = drv.Open(point, 0)
    point_lyr = point.GetLayer()
    polygon = drv.Open(polygon, 0)
    polygon_lyr = polygon.GetLayer()
    union1 = ogr.Geometry(3)
    for feature in polygon_lyr:
       geom = feature.GetGeometryRef()
       union1 = union1.Union(geom)
    union2 = ogr.Geometry(point_lyr.GetGeomType())
    for feat in point_lyr:
       geom = feat.GetGeometryRef()
       union2 = union2.Union(geom)
    intersect = union1.Intersection(union2)
    dstshp = drv.CreateDataSource(out)
    dstlayer = dstshp.CreateLayer(out, geom_type=ogr.wkbPoint)
    featureDefn = dstlayer.GetLayerDefn()
    for point in intersect:
       feature = ogr.Feature(featureDefn)
       coord = str.split(str(point), " ")
       x = float(coord[1][1:])
       y = float(coord[2][:-1])
       ponto = ogr.Geometry(ogr.wkbPoint)
       ponto.AddPoint(x, y)
       feature.SetGeometry(ponto)
       dstlayer.CreateFeature(feature)
def cria_amostras_classe(entrada, fsetenta, ftrinta, classe):
    drv = ogr.GetDriverByName("ESRI SHAPEFILE")
    am_classe = drv.Open(entrada, 0)
    lyr = am_classe.GetLayer()
    nr_pontos = 0
    for ls in lyr:
       nr_pontos += 1
    if nr_pontos < 4:
       return
    setenta = int((nr_pontos * 70) / 100)
    del am_classe, lyr
    am_classe = drv.Open(entrada, 0)
    lyr = am_classe.GetLayer()
    dic70 = []
    dic30 = []
    conta = 1
    for feature in lyr:
       geom = feature.GetGeometryRef()
       coord = str.split(str(geom), " ")
       x = float(coord[1][1:])
       y = float(coord[2][:-1])
       temp = [x,y]
```

```
        if conta <= setenta:
            dic70.append(temp)
        else:
            dic30.append(temp)
        conta += 1
    dstshp70 = drv.CreateDataSource(fsetenta + "\\setenta_" + str(classe) + ".shp")
    dstlayer70 = dstshp70.CreateLayer(fsetenta + "\\setenta_" + str(classe) + ".shp",
    geom_type=ogr.wkbPoint)
    featureDefn = dstlayer70.GetLayerDefn()
    for coord in dic70:
        feature = ogr.Feature(featureDefn)
        ponto = ogr.Geometry(ogr.wkbPoint)
        ponto.AddPoint(coord[0], coord[1])
        feature.SetGeometry(ponto)
        dstlayer70.CreateFeature(feature)
    dstshp30 = drv.CreateDataSource(ftrinta + "\\trinta_" + str(classe) + ".shp")
    dstlayer30 = dstshp30.CreateLayer(ftrinta + "\\trinta_" + str(classe) + ".shp",
    geom_type=ogr.wkbPoint)
    featureDefn = dstlayer30.GetLayerDefn()
    for coord in dic30:
        feature = ogr.Feature(featureDefn)
        ponto = ogr.Geometry(ogr.wkbPoint)
        ponto.AddPoint(coord[0], coord[1])
        feature.SetGeometry(ponto)
        dstlayer30.CreateFeature(feature)
def criar_amostras_factor(fsetenta, ftrinta, conta, dsetenta, dtrinta):
    lst_setenta = lista_shapefile(fsetenta)
    lst_trinta = lista_shapefile(ftrinta)
    drv = ogr.GetDriverByName("ESRI SHAPEFILE")
    lista = []
    for i in lst_setenta:
        shp = drv.Open(i, 0)
        lyr = shp.GetLayer()
        for feature in lyr:
            geom = feature.GetGeometryRef()
            coord = str.split(str(geom), " ")
            x = float(coord[1][1:])
            y = float(coord[2][:-1])
            t = [x, y]
            lista.append(t)
    dstshp70 = drv.CreateDataSource(dsetenta + "\\factor70_" + str(conta) + ".shp")
    dstlayer70 = dstshp70.CreateLayer(dsetenta + "\\factor70_" + str(conta) + ".shp",
    geom_type=ogr.wkbPoint)
    featureDefn = dstlayer70.GetLayerDefn()
    for coord in lista:
        feature = ogr.Feature(featureDefn)
        ponto = ogr.Geometry(ogr.wkbPoint)
        ponto.AddPoint(coord[0], coord[1])
        feature.SetGeometry(ponto)
        dstlayer70.CreateFeature(feature)
    del lista
    lista = []
    for i in lst_trinta:
        shp = drv.Open(i, 0)
        lyr = shp.GetLayer()
        for feature in lyr:
            geom = feature.GetGeometryRef()
            coord = str.split(str(geom), " ")
            x = float(coord[1][1:])
            y = float(coord[2][:-1])
            t = [x, y]
            lista.append(t)
    dstshp30 = drv.CreateDataSource(dtrinta + "\\factor30_" + str(conta) + ".shp")
```

```
    dstlayer30 = dstshp30.CreateLayer(dtrinta + "\\factor30_" + str(conta) + ".shp",
    geom_type=ogr.wkbPoint)
    featureDefn = dstlayer30.GetLayerDefn()
    for coord in lista:
       feature = ogr.Feature(featureDefn)
       ponto = ogr.Geometry(ogr.wkbPoint)
       ponto.AddPoint(coord[0], coord[1])
       feature.SetGeometry(ponto)
       dstlayer30.CreateFeature(feature)
def main_job_Gdal(fenomeno, w_factores, saida):
    prj = fenomeno[:-4] + ".prj"
    gdal.UseExceptions()
    lst_factores = lista_factores(w_factores)
    os.mkdir("C:\\rst2shp")
    conta = 0
    pastas = []
    for i in lst_factores:
       rst2Polygon(i, "C:\\rst2shp\\factor_" + str(conta) + ".shp")
       cria_prj(prj, "C:\\rst2shp\\factor_" + str(conta) + ".prj")
       gdal_dissolve_split("C:\\rst2shp\\factor_" + str(conta) + ".shp", "value", conta)
       pastas.append("C:\\factor_" + str(conta))
       conta += 1
    areatrab = "C:\\areatrab"
    os.mkdir(areatrab)
    caminho_setenta_factor = areatrab + "\\factor_setenta"
    caminho_trinta_factor = areatrab + "\\factor_trinta"
    os.mkdir(caminho_setenta_factor)
    os.mkdir(caminho_trinta_factor)
    id_factor = 0
    for pasta in pastas:
       caminho_setenta_classe = areatrab + "\\classe_70" + str(id_factor)
       caminho_trinta_classe = areatrab + "\\classe_30" + str(id_factor)
       os.mkdir(caminho_setenta_classe)
       os.mkdir(caminho_trinta_classe)
       logos = 0
       lst = lista_shapefile(pasta)
       os.mkdir("C:\\temporario" + str(id_factor))
       for shp in lst:
          intersection(fenomeno, shp, "C:\\temporario" + str(id_factor) + "\\am_classe_" +
          str(logos) + ".shp")
          cria_amostras_classe("C:\\temporario" + str(id_factor) + "\\am_classe_" +
          str(logos) + ".shp", caminho_setenta_classe, caminho_trinta_classe, logos)
          logos += 1
       criar_amostras_factor(caminho_setenta_classe, caminho_trinta_classe, id_factor,
       caminho_setenta_factor, caminho_trinta_factor)
       id_factor += 1
    del id_factor, logos
    excel = xlwt.Workbook()
    for amostra in range(len(lst_factores)):
       nr_linha = 0
       add_livro = excel.add_sheet("amostra_" + str(amostra))
       amostra_70 = caminho_setenta_factor + "\\factor70_" + str(amostra) + ".shp"
       amostra_30 = caminho_trinta_factor + "\\factor30_" + str(amostra) + ".shp"
       for pasta in range(len(pastas)):
          novo = areatrab + "\\temp" + str(amostra) + "_" + str(pasta)
          os.mkdir(novo)
          add_livro.write(nr_linha, 0, pastas[pasta])
          nr_linha += 1
          lst_shp = lista_shapefile(pastas[pasta])
          add_livro.write(nr_linha, 1, 'ponto_70')
          add_livro.write(nr_linha, 2, 'ponto_30')
          conta = 0
          nr_linha += 1
          for classe in range(len(lst_shp)):
```

```
            intersection(amostra_70, lst_shp[classe], novo + "\\intersect70_"+ str(conta) +
            ".shp")
            drv = ogr.GetDriverByName("ESRI SHAPEFILE")
            shp = drv.Open(novo + "\\intersect70_" + str(conta) + ".shp", 0)
            lyr = shp.GetLayer()
            conta_70 = 0
            for feat in lyr:
               conta_70 += 1
            intersection(amostra_30, lst_shp[classe], novo + "\\intersect30_"+ str(conta) +
            ".shp")
            shp = drv.Open(novo + "\\intersect30_" + str(conta) + ".shp", 0)
            lyr = shp.GetLayer()
            for feat in lyr:
               conta_30 += 1
            add_livro.write(nr_linha, 0, classe)
            add_livro.write(nr_linha, 1, conta_70)
            add_livro.write(nr_linha, 2, conta_30)
            conta += 1
            nr_linha += 1
    excel.save(saida)
```

Programa 27 - Algoritmo (em Python, v. 2.7.3) que incrementa o Método do Valor Informativo, tendo em vista a obtenção de um cartograma que representa a maior susceptibilidade à ocorrência de movimentos de vertente, segundo um conjunto de factores de predisposição previamente classificados – ArcGIS 10.2 e Arcpy.

```
# Segundo o método do valor informativo, produz um cartograma que representa a
susceptibilidade relativa à ocorrência de movimentos de vertente, em cada
unidade territorial (célula) - ArcGIS
import arcpy, unicodedata, os
movimentos = arcpy.GetParameterAsText(0)
factores = arcpy.GetParameterAsText(1)
risco = arcpy.GetParameterAsText(2)
arcpy.env.overwriteOutput = True
def listar_factores(work_factores):
    arcpy.env.workspace = work_factores
    list_fac = arcpy.ListRasters()
    list_unicode = []
    for i in list_fac:
       e = unicodedata.normalize('NFKD', i).encode('ascii', 'ignore')
       o = work_factores + "\\" + e
       list_unicode.append(o)
    return list_unicode
def cria_layer(objecto, nome_layer, rst_ou_vect):
    if rst_ou_vect == True:
       lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
       return lyr
    else:
       lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
       return lyr
def obtem_cellsize(raster):
    lyr = cria_layer(raster, "rst", True)
    c = str(arcpy.GetRasterProperties_management(lyr, "CELLSIZEX", "Band_1"))
    del lyr
    return c
def nr_pixel_areaEstudo(raster):
    lyr = cria_layer(raster, "rst", True)
    cs = arcpy.SearchCursor(lyr)
    ls = cs.next()
    nr_pixeis = 0
```

```
    while ls:
       count = ls.getValue("Count")
       nr_pixeis += count
       ls = cs.next()
    del count, ls, cs, lyr
    return nr_pixeis
def nr_pixel_areaMovimentos(ocorrencias, saida, resolucao):
    arcpy.PolygonToRaster_conversion(ocorrencias, "FID", saida, "CELL_CENTER", "NONE",
    str(resolucao))
    cs = arcpy.SearchCursor(saida)
    ls = cs.next()
    nr_pixeis = 0
    while ls:
       count = ls.getValue("Count")
       nr_pixeis += count
       ls = cs.next()
    del count, ls, cs
    return nr_pixeis
def rst2Polygon(entrada, saida):
    arcpy.RasterToPolygon_conversion(entrada, saida, "NO_SIMPLIFY", "Value")
def agregar(entrada, saida):
   lyr = cria_layer(entrada, "lyr", False)
   arcpy.Dissolve_management(lyr, saida, "GRIDCODE", "", "MULTI_PART", "")
def criar_campo(rst):
    if momento == 1:
       arcpy.AddField_management(rst, "VInfo", "FLOAT", "", "", "", "", "NON_NULLABLE",
       "NON_REQUIRED", "")
    elif momento == 2:
       arcpy.AddField_management(rst, "lnVI", "FLOAT", "", "", "", "", "NON_NULLABLE",
       "NON_REQUIRED", "")
def valor_informativo(factor, ocorrencias, resolucao, template):
    arcpy.env.workspace = "C:\\temporario"
    lyr = cria_layer(factor, "factor", False)
    cs = arcpy.SearchCursor(lyr)
    ls = cs.next()
    dic = {}
    while ls:
       classe = int(ls.getValue("GRIDCODE"))
       feature_layer = "classe_" + str(classe) + ".shp"
       expressao = "GRIDCODE = " + str(classe)
       arcpy.Select_analysis(lyr, feature_layer, expressao)
       clip = "clip_fenomeno_" + str(classe) + ".shp"
       arcpy.Intersect_analysis([feature_layer, ocorrencias], clip, "", "", "")
       try:
          arcpy.env.snapRaster = template
          arcpy.PolygonToRaster_conversion("C:\\temporario\\" + clip,
          "FID","rst_ocorrencia_" + str(classe) + ".tif", "CELL_CENTER", "NONE",
          str(resolucao))
          c = arcpy.SearchCursor("C:\\temporario\\rst_ocorrencia_" + str(classe) + ".tif")
          l = c.next()
          count_pixeis_fenomeno = 0
          while l:
             nr_celulas = int(l.getValue("Count"))
             count_pixeis_fenomeno += nr_celulas
             l = c.next()
          del c, l, nr_celulas
          dic.update({int(classe):count_pixeis_fenomeno})
       except:
          dic.update({int(classe):0})
       ls = cs.next()
    return dic
def main_ArcGIS(fenomeno, wfactores, output):
    lst_factores = listar_factores(wfactores)
    cellsize = obtem_cellsize(lst_factores[0])
```

```
    area_total = nr_pixel_areaEstudo(lst_factores[0])
    lyr_fenomeno = cria_layer(fenomeno, "movimentos", False)
    areatrab = "C:\\areatrab"
    os.mkdir(areatrab)
    area_ocorrencias = nr_pixel_areaMovimentos(movimentos, areatrab +
    "\\rst_fenomeno.tif", cellsize)
    os.mkdir("C:\\temporario")
    conta = 0
    for i in lst_factores:
       rst_layer = cria_layer(i, "rst_" + str(conta), True)
       criar_campo(rst_layer, 1)
       criar_campo(rst_layer, 2)
       rst2Polygon(rst_layer, "C:\\temporario\\factor_" + str(conta) + ".shp")
       agregar("C:\\temporario\\factor_" + str(conta) + ".shp", "C:\\areatrab\\dfactor_"
       + str(conta) + ".shp")
       vi = valor_informativo("C:\\areatrab\\dfactor_" + str(conta) + ".shp", lyr_fenomeno,
       cellsize, lst_factores[0])
       up = arcpy.UpdateCursor(rst_layer)
       import math
       for u in up:
          id_classe = int(u.getValue("Value"))
          count_classe = u.getValue("Count")
          count_ocorrencia = vi[id_classe]
          val_info = (float(count_ocorrencia) / count_classe) / (float(area_ocorrencias) /
          area_total)
          if val_info == 0:
             lnvi = -2
          else:
             lnvi = math.log10(val_info)
          u.setValue("VInfo", val_info)
          u.setValue("lnVI", lnvi)
          up.updateRow(u)
       arcpy.env.workspace = "C:\\areatrab"
       entrada = i + " lnVI 1"
       saida = "var_" + str(conta) + ".tif"
       tempEnvironment0 = arcpy.env.extent
       arcpy.env.extent = lst_factores[0]
       arcpy.gp.WeightedSum_sa(entrada, saida)
       arcpy.env.extent = tempEnvironment0
       conta += 1
    del conta, rst_layer
    for i in range(len(lst_factores)):
       lyr = cria_layer("C:\\areatrab\\var_" + str(i) + ".tif", "rst_" + str(i), True)
       expressao_temp = "\"" + "rst_" + str(i) + "\""
       if i == 0:
          expressao = expressao_temp + "+"
       else:
          expressao = expressao + "+" + expressao_temp
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = lst_factores[0]
    arcpy.gp.RasterCalculator_sa(expressao, output)
    arcpy.env.extent = tempEnvironment0
main_ArcGIS(movimentos, factores, risco)
```

Programa 28 - Algoritmo (em Python, v. 2.7.4) que incrementa o Método do Valor Informativo, tendo em vista a obtenção de um cartograma que representa a maior susceptibilidade à

ocorrência de movimentos de vertente, segundo um conjunto de factores de predisposição previamente classificados – GRASS GIS 7.0.0 e PyGRASS.

```
# Segundo o método do valor informativo, produz um cartograma que representa a
suscepetibilidade relativa à ocorrência de movimentos de vertente, em cada
unidade territorial (célula) - GRASS GIS
def r_in_gdal(raster, grass_raster):
    from grass.pygrass.modules import Module
    adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
    overwrite=True, run_=False)
    adiciona()
def lista_factores(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
       r.extend(filenames)
       break
    t = []
    for i in r:
       ext = i[-4:]
          if ext == ".tif":
             t.append(i)
    f = []
    for i in t:
       r_in_gdal(path + "\\" + i, i[:-4])
       f.append(i[:-4])
    return f
def db_in_ogr(tabela, nome):
    from grass.pygrass.modules import Module
    adiciona = Module("db.in.ogr", input=tabela, output=nome, run_=False)
    adiciona()
def obter_resolucao(rst):
    import grass.script as grass
    dic = grass.raster.raster_info(rst)
    for chave in dic.keys():
       if chave == "nsres":
          valor = dic[chave]
    return valor
def cria_objTable(tab, path):
    from grass.pygrass.vector.table import Table, get_path
    import sqlite3
    objTabela = Table(name=tab, connection=sqlite3.connect(get_path(path)))
    return objTabela
def conta_celula(tabela, grassdata_workspace, location, mapset):
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
    "\\sqlite\\sqlite.db"
    tabela = cria_objTable(tabela, caminho)
    conta = 0
    for linha in tabela:
       conta += int(linha[1])
    return conta
def v_in_ogr(shapefile, grass_vector):
    from grass.pygrass.modules import Module
    adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
    overwrite=True, run_=False)
    adiciona()
def v2rst(vector, raster):
    from grass.pygrass.modules import Module
    f = Module("v.to.rast", input=vector, type='area', use='cat', output=raster,
    overwrite=True, run_=False)
    f()
def conta_pixel_ocorrencias(r):
```

```
    from grass.pygrass import raster
    fe = raster.RasterRow(r)
    fe.open()
    c = 0
    for row in fe:
       for i in row:
          if i < 0.0 or i > 500:
             continue
          else:
             c += 1
    return c
def RasterToPolygon(raster, vector):
    from grass.pygrass.modules import Module
    raster2poly = Module("r.to.vect", input=raster, output=vector, type='area',
    column='value', overwrite=True, run_=False)
    raster2poly()
def id_classes(in_vector, field, grassdata_workspace, location, mapset):
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
    "\\sqlite\\sqlite.db"
    tabela = cria_objTable(in_vector, caminho)
    cls = []
    for linha in tabela:
       cls.append(int(linha[1]))
    import numpy
    cls = numpy.unique(cls)
    t = []
    for i in cls:
       if i < 0 or i > 20:
          continue
       else:
          t.append(i)
    return t
def extract(vector, name_output, classe):
    from grass.pygrass.modules import Module
    onde = "value = " + str(classe)
    extrai = Module("v.extract", input=vector, type="area", where=onde,
    output=name_output, overwrite=True, run_=False)
    extrai()
def intersection(aentrada, bentrada, saida):
    from grass.pygrass.modules import Module
    clip = Module("v.overlay", ainput=aentrada, atype="area", binput=bentrada,
    btype="area", operator="and", output=saida, overwrite=True, run_=False)
    clip()
def valor_informativo(classes, lst_extract, fenomeno, factor):
    c = 0
    dic = {}
    for classe in range(len(classes)):
       intersection(fenomeno, lst_extract[classe], "mov_" + str(factor) + "_" + str(c))
       v2rst("mov_" + str(factor) + "_" + str(c), "r_mov_" + str(factor) + "_" + str(c))
       count_pixeis_fenomeno = conta_pixel_ocorrencias("r_mov_" + str(factor) +"_" + str(c))
       dic.update({int(classes[classe]):count_pixeis_fenomeno})
       c += 1
    return dic
def nr_pixel_classe(tabela, grassdata_workspace, location, mapset):
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
    "\\sqlite\\sqlite.db"
    tabela = cria_objTable(tabela, caminho)
    dic = {}
    for linha in tabela:
       temp = {int(linha[0]):int(linha[1])}
       dic.update(temp)
    return dic
def reclass(raster, saida, dic_vi):
```

```
    import os
    os.mkdir("C:\\temporario")
    txt = open("C:\\temporario\\regras.txt", "w")
    for chave in dic_vi.keys():
       txt.write(str(chave) + "  = " + str(dic_vi[chave][1]) + "\n")
    txt.close()
    regras = 'C:\\temporario\\regras.txt'
    from grass.pygrass.modules import Module
    reclass = Module("r.reclass", input=raster, output=saida, rules=regras,
    overwrite=True, run_=False)
    reclass()
    import shutil
    shutil.rmtree("C:\\temporario")
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
def main_GRASS(fenomeno, wfactores, output, wgrass, location, mapset):
    lst_factores = lista_factores(wfactores)
    lst_table = []
    for i in lst_factores:
       db_in_ogr(wfactores + "\\" + i + ".tif.vat.dbf", "db_" + i)
       lst_table.append("db_" + i)
    cellsize = float(obter_resolucao(lst_factores[0]))
    area_total = conta_celula(lst_table[0], wgrass, location, mapset)
    v_in_ogr(fenomeno, "movimentos")
    v2rst("movimentos", "r_movimentos")
    area_ocorrencias = conta_pixel_ocorrencias("r_movimentos")
    conta = 0
    lst_reclss = []
    for i in range(len(lst_factores)):
       RasterToPolygon(lst_factores[i], "v_" + lst_factores[i])
       classes = id_classes("v_" + lst_factores[i], "value", wgrass, location, mapset)
       lextract = []
       for cls in classes:
          extract("v_" + lst_factores[i], "e_" + lst_factores[i] + "_" + str(cls), cls)
          lextract.append("e_" + lst_factores[i] + "_" + str(cls))
       vi = valor_informativo(classes, lextract, "movimentos", conta)
       _vi_ = {}
       px_classe = nr_pixel_classe(lst_table[i], wgrass, location, mapset)
       import math
       for chave in px_classe:
          id_classe = int(chave)
          count_classe = px_classe[chave]
          count_ocorrencia = vi[id_classe]
          val_info = (float(count_ocorrencia) / count_classe) / (float(area_ocorrencias) /
          area_total)
          if val_info == 0:
             lnvi = -2 * 1000000
          else:
             lnvi = ((math.log10(val_info)) * 1000000)
          _vi_.update({id_classe:[val_info, lnvi]})
       reclass(lst_factores[i], "rcls_" + lst_factores[i], _vi_)
       lst_reclss.append("rcls_" + lst_factores[i])
       conta += 1
    c = 0
    for i in lst_reclss:
       if c == 0:
          expressao = i
       else:
          expressao = expressao + "+" + i
       c += 1
    from grass.pygrass.modules import Module
```

```
    algebra = Module("r.mapcalc", "susceptibilidade = " + expressao, overwrite=True,
    run_=False)
    algebra()
    r_out_gdal("susceptibilidade", output)
```

Programa 29 - Algoritmo (em Python, v. 2.7.5) que incrementa o Método do Valor Informativo, tendo em vista a obtenção de um cartograma que representa a maior susceptibilidade à ocorrência de movimentos de vertente, segundo um conjunto de factores de predisposição previamente classificados – QGIS 2.10.1 e qgis.core e processing.

```
# Segundo o método do valor informativo, produz um cartograma que representa a
suscepetibilidade relativa à ocorrência de movimentos de vertente, em cada
unidade territorial (célula) – QGIS 2.10.1
import os, shutil, processing
from qgis.core import QgsRasterLayer, QgsVectorLayer
from osgeo import gdal, ogr
##MovVert= group
##ValInformativo= name
##movimentos= vector
##factores= folder
##susceptibilidade= output raster
def discrimina_factores(caminho):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(caminho):
       r.extend(filenames)
       break
    d = {}
    c = 0
    for i in r:
       ext =  i[-4:]
       if ext == ".tif":
          d.update({c:[caminho + "\\" + i, caminho + "\\" + i + ".vat.dbf"]})
          c += 1
    return d
def cria_layer(objecto, nome, rst_ou_vect):
    if rst_ou_vect == True:
       rst_lyr = QgsRasterLayer(objecto, nome)
       return rst_lyr
    elif rst_ou_vect == False:
       v_lyr = QgsVectorLayer(objecto, nome, "ogr")
       return v_lyr
def obtem_cellsize(rasterLyr):
    x = rasterLyr.rasterUnitsPerPixelX()
    return x
def v2rst(vector, raster):
    processing.runalg("gdalogr:rasterize", vector, "ID", 1, 10, 10, 1, "-9999", 4, None,
    6, 1, False, 0, True, raster)
def conta_pixel(raster):
    img = gdal.Open(raster)
    array = img.ReadAsArray()
    c = 0
    for i in array:
       for e in i:
          if e < 0 or e > 500:
             continue
          else:
             c += 1
    return c
def rst2polygon(entrada, saida):
```

```
    processing.runalg("gdalogr:polygonize", entrada, "value", saida)
def split(entrada, workspace_saida):
    processing.runalg("qgis:splitvectorlayer", entrada, "value", workspace_saida)
def lista_shapefile(path):
    from os import walk
    r = []
    for (dirpath, dirnames, filenames) in walk(path):
       r.extend(filenames)
       break
    t = []
    for i in r:
       ext = i[-4:]
       if ext == '.shp':
          t.append(path + "\\" + i)
    return t
def le_tabela(tabela_raster):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(tabela_raster, 0)
    layer = dataSource.GetLayer()
    conta = 0
    for linha in layer:
       count = int(linha.GetField('Count'))
       conta += count
    return conta
def id_classes(tabela):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(tabela, 0)
    layer = dataSource.GetLayer()
    lst = []
    for linha in layer:
       classe = int(linha.GetField("Value"))
       lst.append(classe)
    return lst
def intersection(poligonos1, poligonos, saida):
    processing.runalg("qgis:intersection", poligonos1, poligonos, saida)
def valor_informativo(dic, fenomeno, factor, workspace):
    c = 0
    d = {}
    for chave in dic.keys():
       intersection(fenomeno, dic[chave], workspace + "\\mov_" + str(factor) +"_" + str(c) +
       ".shp")
       layer = cria_layer(workspace + "\\mov_" + str(factor) + "_" + str(c) + ".shp",
       "layer", False)
       v2rst(layer, workspace + "\\r_mov_" + str(factor) + "_" + str(c) + ".tif")
       try:
          count_pixeis_fenomeno = conta_pixel(workspace + "\\r_mov_" + str(factor) + "_" +
          str(c) + ".tif")
       except:
          count_pixeis_fenomeno = 0
       d.update({int(chave):count_pixeis_fenomeno})
       c += 1
    return d
def nr_pixel_classe(table_raster):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(table_raster, 0)
    layer = dataSource.GetLayer()
    dic = {}
    for linha in layer:
       temp = {int(linha.GetField('Value')):int(linha.GetField('Count'))}
       dic.update(temp)
    return dic
def reclass(raster, saida, dic_vi):
    os.mkdir("C:\\regras")
```

```
    txt = open("C:\\regras\\regras.txt", "w")
    for chave in dic_vi.keys():
       txt.write(str(chave) + "  = " + str(dic_vi[chave][1]) + "\n")
    txt.close()
    regras = "C:\\regras\\regras.txt"
    processing.runalg("grass:r.reclass",raster, regras, None, 0.0, saida)
    shutil.rmtree("C:\\regras")
def ass_cls_split(id_cls, l_classe):
    dic = {}
    for classe in id_cls:
       for shp in l_classe:
          l = shp[-6:]
          nr = l[:2]
          if "_" + str(classe) == nr or str(classe) == nr:
             dic.update({classe:shp})
    return dic
def main_vi_QGIS(fenomeno, wfactores, output):
    d_factores = discrimina_factores(wfactores)
    cellsize = float(obtem_cellsize(cria_layer(d_factores[0][0], "lyr", True)))
    area_total = le_tabela(d_factores[0][1])
    w = "C:\\areatrab"
    os.mkdir(w)
    lyr_fenomeno = cria_layer(fenomeno, "lyr_fenomeno", False)
    v2rst(lyr_fenomeno, w +  "\\r_movimentos.tif")
    area_ocorrencias = conta_pixel(w +  "\\r_movimentos.tif")
    lst_reclss = []
    for chave in d_factores.keys():
       os.mkdir("C:\\dfactor" + str(chave))
       rst2polygon(d_factores[chave][0], "C:\\dfactor" + str(chave) + "\\factor_" +
       str(chave) + ".shp")
       split("C:\\dfactor" + str(chave) + "\\factor_" + str(chave) + ".shp",
       "C:\\dfactor" + str(chave))
       classes = id_classes(d_factores[chave][1])
       lst_classes = lista_shapefile("C:\\dfactor" + str(chave))
       dic_cls = ass_cls_split(classes, lst_classes)
       vi = valor_informativo(dic_cls, lyr_fenomeno, chave, w)
       _vi_ = {}
       px_classe = nr_pixel_classe(d_factores[chave][1])
       import math
       for chave in px_classe.keys():
          id_classe = int(chave)
          count_classe = px_classe[chave]
          count_ocorrencia = vi[id_classe]
          val_info = (float(count_ocorrencia) / count_classe) / (float(area_ocorrencias) /
          area_total)
          if val_info == 0:
             lnvi = -2 * 1000000
          else:
             lnvi = ((math.log10(val_info)) * 1000000)
          _vi_.update({id_classe:[val_info, lnvi]})
       reclass(d_factores[chave][0], w + "\\reclss_factor_" + str(chave) + ".tif", _vi_)
       lst_reclss.append(w + "\\reclss_factor_" + str(chave) + ".tif")
    processing.runalg("grass:r.mapcalculator", lst_reclss[0], lst_reclss[1],
    lst_reclss[2], lst_reclss[3], lst_reclss[4], lst_reclss[5], "A+B+C+D+E+F", None,
    0.0, output)
main_vi_QGIS(movimentos, factores, susceptibilidade)
```

Programa 30 - Algoritmo (em Python, v. 2.7.3) que incrementa o Método do Valor Informativo, tendo em vista a obtenção de um cartograma que representa a maior susceptibilidade à

ocorrência de movimentos de vertente, segundo um conjunto de factores de predisposição previamente classificados – GDAL 1.11.1.

```
# Segundo o método do valor informativo, produz um cartograma que representa a
suscepetibilidade relativa à ocorrência de movimentos de vertente, em cada
unidade territorial (célula) – GDAL
import os, shutil, numpy, math
from osgeo import gdal, ogr, osr
def discrimina_factores(caminho):
      from os import walk
      r = []
      for (dirpath, dirnames, filenames) in walk(caminho):
         r.extend(filenames)
         break
      d = {}
      c = 0
      for i in r:
         ext =  i[-4:]
         if ext == ".tif":
            d.update({c:[caminho + "\\" + i, caminho + "\\" + i + ".vat.dbf"]})
            c += 1
      return d
def ler_tabela(t):
      driver = ogr.GetDriverByName("ESRI Shapefile")
      dataSource = driver.Open(t, 0)
      lyr = dataSource.GetLayer()
      z = 0
      for l in lyr:
         c = int(l.GetField("Count"))
         z += c
      return z
def cria_prj(f_prj, saida):
      texto = open(f_prj, 'r')
      for linha in texto.readlines():
         coord = linha
      texto.close()
      new_prj = open(saida, "w")
      new_prj.write(str(coord))
      new_prj.close()
def obtem_cellsize(raster):
      img = gdal.Open(raster)
      (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) =
      img.GetGeoTransform()
      return x_size
def shp2rst(entrada, resolucao, NoDataValue, saida):
      source_ds = ogr.Open(entrada)
      source_layer = source_ds.GetLayer()
      x_min, x_max, y_min, y_max = source_layer.GetExtent()
      x_res = int((x_max - x_min) / resolucao)
      y_res = int((y_max - y_min) / resolucao)
      target = gdal.GetDriverByName("GTiff").Create(saida, x_res, y_res, 1,
      gdal.GDT_UInt16)
      try:
         target.SetGeoTransform((x_min, resolucao, 0, y_max, 0, -resolucao))
         band = target.GetRasterBand(1)
         band.SetNoDataValue(NoDataValue)
         gdal.RasterizeLayer(target, [1], source_layer)
      except:
         return
def conta_ocorrencias(rst):
      img = gdal.Open(rst)
      array = numpy.array(img.ReadAsArray())
      n = 0
```

```
      for i in array:
         for e in i:
            if e == 255:
               n += 1
      return n
def rst2Polygon(entrada, saida):
      img = gdal.Open(entrada)
      band = img.GetRasterBand(1)
      drv = ogr.GetDriverByName("ESRI SHAPEFILE")
      shp = drv.CreateDataSource(saida)
      shp_lyr = shp.CreateLayer(saida, srs=None)
      shp_lyr.CreateField(ogr.FieldDefn("value", ogr.OFTInteger))
      gdal.Polygonize(band, None, shp_lyr, 0, [], callback=None)
def split_gdal(entrada, w, classe, sre):
      inDriver = ogr.GetDriverByName("ESRI SHAPEFILE")
      inDataSource = inDriver.Open(entrada, 0)
      inLayer = inDataSource.GetLayer()
      inLayer.SetAttributeFilter("value = " + str(classe))
      outDriver = ogr.GetDriverByName("ESRI SHAPEFILE")
      outDataSource = outDriver.CreateDataSource(w + "\\parte_" + str(classe) + ".shp")
      outLayer_name = "parte_" + str(classe)
      outLayer = outDataSource.CreateLayer(outLayer_name, geom_type=ogr.wkbPolygon)
      inLayerDefn = inLayer.GetLayerDefn()
      for i in range(0, inLayerDefn.GetFieldCount()):
         fieldDefn = inLayerDefn.GetFieldDefn(i)
         fieldName = fieldDefn.GetName()
         outLayer.CreateField(fieldDefn)
      outLayerDefn = outLayer.GetLayerDefn()
      for inFeature in inLayer:
         outFeature = ogr.Feature(outLayerDefn)
         for i in range(0, outLayerDefn.GetFieldCount()):
            fieldDefn = outLayerDefn.GetFieldDefn(i)
            fieldName = fieldDefn.GetName()
            outFeature.SetField(outLayerDefn.GetFieldDefn(i).GetNameRef(),inFeature.GetField(
            i))
         geom = inFeature.GetGeometryRef()
         outFeature.SetGeometry(geom.Clone())
         outLayer.CreateFeature(outFeature)
      inDataSource.Destroy()
      outDataSource.Destroy()
      cria_prj(sre, w + "\\parte_" + str(classe) + ".prj")
def id_classes(tabela):
      driver = ogr.GetDriverByName("ESRI Shapefile")
      dataSource = driver.Open(tabela, 0)
      layer = dataSource.GetLayer()
      lst = []
      for linha in layer:
         classe = int(linha.GetField("Value"))
         lst.append(classe)
      return lst
def lista_shapefile(path):
      from os import walk
      r = []
      for (dirpath, dirnames, filenames) in walk(path):
         r.extend(filenames)
         break
      t = []
      for i in r:
         ext = i[-4:]
         if ext == '.shp':
            t.append(path + "\\" + i)
      return t
def ass_cls_split(id_cls, l_classe):
      dic = {}
```

```
    for classe in id_cls:
        for shp in l_classe:
            l = shp[-6:]
            nr = l[:2]
            if "_" + str(classe) == nr or str(classe) == nr:
                dic.update({classe:shp})
    return dic
def nr_pixel_classe(table_raster):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(table_raster, 0)
    layer = dataSource.GetLayer()
    dic = {}
    for linha in layer:
        temp = {int(linha.GetField('Value')):int(linha.GetField('Count'))}
        dic.update(temp)
    return dic
def intersection(point, polygon, out):
    drv = ogr.GetDriverByName("ESRI SHAPEFILE")
    point = drv.Open(point, 0)
    point_lyr = point.GetLayer()
    polygon = drv.Open(polygon, 0)
    polygon_lyr = polygon.GetLayer()
    union1 = ogr.Geometry(3)
    for feature in polygon_lyr:
        geom = feature.GetGeometryRef()
        union1 = union1.Union(geom)
    union2 = ogr.Geometry(point_lyr.GetGeomType())
    for feat in point_lyr:
        geom = feat.GetGeometryRef()
        union2 = union2.Union(geom)
    intersect = union1.Intersection(union2)
    dstshp = drv.CreateDataSource(out)
    dstlayer = dstshp.CreateLayer(out, geom_type=ogr.wkbPolygon)
    featureDefn = dstlayer.GetLayerDefn()
    i_poligonos = {}
    id_feature = 0
    for polygon in intersect:
        figura = str(polygon)
        l_polygon = str.split(figura[10:-2], ",")
        i_poligonos.update({id_feature:l_polygon})
        id_feature +=1
    for chave in i_poligonos.keys():
        feature = ogr.Feature(featureDefn)
        ring = ogr.Geometry(ogr.wkbLinearRing)
        for coord in i_poligonos[chave]:
            coord = str.split(coord, " ")
            try:
              x = float(coord[0])
              y = float(coord[1])
            except:
              x = float(coord[1][1:])
              y = float(coord[2])
            ring.AddPoint(x, y)
        poly = ogr.Geometry(ogr.wkbPolygon)
        poly.AddGeometry(ring)
        feature.SetGeometry(poly)
        dstlayer.CreateFeature(feature)
def pre_valor_informativo(dic, fenomeno, factor, workspace, resolucao, sre):
    c = 0
    d = {}
    for chave in dic.keys():
        intersection(fenomeno, dic[chave], workspace + "\\mov_" + str(factor) +"_" + str(c)
        + ".shp")
        cria_prj(sre, workspace + "\\mov_" + str(factor) + "_" + str(c) + ".prj")
```

```
        shp2rst(workspace + "\\mov_" + str(factor) + "_" + str(c) + ".shp", resolucao, -
        9999, workspace + "\\r_mov_" + str(factor) + "_" + str(c) +".tif")
        try:
           count_pixeis_fenomeno = conta_ocorrencias(workspace + "\\r_mov_" + str(factor) +
           "_" + str(c) + ".tif")
        except:
           count_pixeis_fenomeno = 0
        d.update({int(chave):count_pixeis_fenomeno})
        c += 1
        return d
def array2raster(entrada, saida, width, height, vector, epsg):
     img = gdal.Open(entrada)
     (upper_left_x, x_size, x_rotation, upper_left_y, y_rotation, y_size) =
     img.GetGeoTransform()
     cols = vector.shape[1]
     rows = vector.shape[0]
     originX = upper_left_x
     originY = upper_left_y
     driver = gdal.GetDriverByName("GTiff")
     outRst = driver.Create(saida, cols, rows, 1, gdal.GDT_Int32)
     outRst.SetGeoTransform((originY, width, 0, originX, 0, height))
     outband = outRst.GetRasterBand(1)
     outband.WriteArray(vector)
     outRstSRS = osr.SpatialReference()
     outRstSRS.ImportFromEPSG(epsg)
     outRst.SetProjection(outRstSRS.ExportToWkt())
     outband.FlushCache()
def reclass_vector(raster, dicionario, output, resolucao, sre):
     img = gdal.Open(raster)
     v = numpy.array(img.ReadAsArray())
     new = []
     lst_classes = []
     for chave in dicionario.keys():
        lst_classes.append(chave)
     for i in v:
        linha = []
        for e in i:
           for c in lst_classes:
              if int(e) == int(c):
                 cat = c
              else:
                 continue
           if e < 0 or e > 20:
              continue
           else:
              linha.append(dicionario[cat][1])
        new.append(linha)
     array2raster(raster, output, int(resolucao), int(resolucao), numpy.array(new[::-
     1]), sre)
def create_array(rst):
     img = gdal.Open(rst)
     v = img.ReadAsArray()
     return v
def main_vi_GDAL(fenomeno, wfactores, output, sresp):
     prj = fenomeno[:-4] + ".prj"
     d_factores = discrimina_factores(wfactores)
     area_total = ler_tabela(d_factores[0][1])
     w = "C:\\areatrab"
     os.mkdir(w)
     cellsize = obtem_cellsize(d_factores[0][0])
     shp2rst(fenomeno, cellsize, -9999, w + "\\r_movimentos.tif")
     area_ocorrencias = conta_ocorrencias(w + "\\r_movimentos.tif")
     lst_reclss = []
     for chave in d_factores.keys():
```

```
        rst2Polygon(d_factores[chave][0], w + "\\v_factor_" + str(chave) + ".shp")
        cria_prj(prj, w + "\\v_factor_" + str(chave) + ".prj")
        os.mkdir("C:\\s_factor" + str(chave))
        id_cls = id_classes(d_factores[chave][1])
        for classe in id_cls:
           split_gdal(w + "\\v_factor_" + str(chave) + ".shp", "C:\\s_factor" + str(chave),
           classe, prj)
        lst_files = lista_shapefile("C:\\s_factor" + str(chave))
        dic_cls = ass_cls_split(id_cls, lst_files)
        pre_vi = pre_valor_informativo(dic_cls, fenomeno, chave, w, cellsize, prj)
        _vi_ = {}
        px_classe = nr_pixel_classe(d_factores[chave][1])
        for ch in px_classe.keys():
           id_classe = int(ch)
           count_classe = px_classe[ch]
           count_ocorrencia = pre_vi[id_classe]
           val_info = (float(count_ocorrencia) / count_classe) / (float(area_ocorrencias) /
           area_total)
           if val_info == 0:
              lnvi = -2 * 1000000
           else:
              lnvi = ((math.log10(val_info)) * 1000000)
           _vi_.update({id_classe:[val_info, lnvi]})
        reclass_vector(d_factores[chave][0], _vi_, w + "\\recls_factor_" + str(chave) +
        ".tif", cellsize, sresp)
        lst_reclss.append(w + "\\recls_factor_" + str(chave) + ".tif")
     vectores = []
     for i in lst_reclss:
        vector = create_array(i)
        vectores.append(vector)
     final = []
     for i in range(len(vectores[0])):
        linha = []
        for e in range(len(vectores[0][i])):
           l_vi =[]
           for u in range(len(vectores)):
              vi = vectores[u][i][e]
              l_vi.append(vi)
           _vi_ = 0
           for vi in l_vi:
              _vi_ += vi
           linha.append(_vi_)
        final.append(linha)
     array2raster(d_factores[0][0], output, cellsize, cellsize, numpy.array(final),
     sresp)
```

Programa 31 - *Script* (em Python, v. 2.7.3) que gera as ausências e extrai os valores das células do *raster* de susceptibilidade que se intersectam com as presenças e ausências, guardando essa informação na tabela de atributos desse tema pontual – ArcGIS 10.2 e ArcPy.

```
# Programa que adiciona as ausências às presenças, extraindo os valores das
células que se intersectam com os pontos (guarda a informação na tabela de
atributos do tema pontual – ArcGIS 10.2
import arcpy, unicodedata, os
arcpy.env.overwriteOutput = True
fenomeno = arcpy.GetParameterAsText(0)
presencas_validacao = arcpy.GetParameterAsText(1)
rst_a_validar = arcpy.GetParameterAsText(2)
lmt = arcpy.GetParameterAsText(3)
def cria_layer(objecto, nome_layer, rst_ou_vect):
    if rst_ou_vect == True:
```

```
        lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
        return lyr
    else:
        lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
        return lyr
def conta_ocorrencias_para_validacao(shp):
    c = arcpy.SearchCursor(shp)
    l = c.next()
    n = 0
    while l:
        n += 1
        l = c.next()
    return n
def cria_pontos_aleatorios(pontos, nr_pontos, lmt):
    arcpy.Buffer_analysis(pontos, "C:\\areatrab\\temp_buffer.shp", "50 Meters", "FULL",
    "ROUND", "NONE", "")
    bf = cria_layer("C:\\areatrab\\temp_buffer.shp", "bf", False)
    l = cria_layer(lmt, "l", False)
    arcpy.Erase_analysis(l, bf, "C:\\areatrab\\shp_queijo.shp")
    arcpy.CreateRandomPoints_management("C:\\areatrab", "nao_fenomeno",
    cria_layer("C:\\areatrab\\shp_queijo.shp", "queijo", False), "", nr_pontos, "50
    Meters", "POINT", "0")
    random = "C:\\areatrab\\nao_fenomeno.shp"
    return random
def cria_campo(lyr):
    arcpy.AddField_management(lyr, "fenom", "SHORT", "", "", "", "", "NULLABLE",
    "NON_REQUIRED", "")
    arcpy.CalculateField_management(lyr, "fenom", "1", "VB", "")
def main_arcGIS(totalidade_ocorrencias, ocorrencias_validacao, susceptibilidade,
limt_estudo):
    w = "C:\\areatrab"
    os.mkdir(w)
    lyr_fenomeno = cria_layer(totalidade_ocorrencias, "lyr_mov", False)
    lyr_validacao = cria_layer(ocorrencias_validacao, "lyr_vali", False)
    nr_pnt = conta_ocorrencias_para_validacao(lyr_validacao)
    ausencias = cria_pontos_aleatorios(lyr_fenomeno, nr_pnt, limt_estudo)
    cria_campo(lyr_validacao)
    arcpy.Append_management(ausencias, ocorrencias_validacao, "NO_TEST", "", "")
    arcpy.sa.ExtractMultiValuesToPoints(ocorrencias_validacao, susceptibilidade,
    "BILINEAR")
main_arcGIS(fenomeno, presencas_validacao, rst_a_validar, lmt)
```

Programa 32 - Bloco de código (em Python, v. 2.7.4) que gera as ausências e extrai os valores das células do *raster* de susceptibilidade que se intersectam com as presenças e ausências, guardando essa informação na tabela de atributos desse tema pontual – GRASS GIS 7.0.0/PyGRASS.

```
# Programa que adiciona as ausências à shapefile que representa a amostra de
validação (que corresponde a 30% das presenças). Tendo as presenças e ausências
no mesmo tema vectorial, o Programa também extrai, do raster que se pretende
validar, os valores das células que se intersectam com os pontos - GRASS GIS
def v_in_ogr(shapefile, grass_vector):
     from grass.pygrass.modules import Module
     adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
     overwrite=True, run_=False)
     adiciona()
def r_in_gdal(raster, grass_raster):
     from grass.pygrass.modules import Module
     adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
     overwrite=True, run_=False)
```

```
      adiciona()
def buffert(shapefile, saida):
      from grass.pygrass.modules import Module
      buffering = Module("v.buffer", input=shapefile, type='point', output=saida,
      distance=50, overwrite=True, run_=False)
      buffering()
def v2rst(vector, raster):
      from grass.pygrass.modules import Module
      f = Module("v.to.rast", input=vector, type='area', use='cat', output=raster,
      overwrite=True, run_=False)
      f()
def union(inputA, inputB, saida):
      from grass.pygrass.modules import Module
      uniao = Module("v.overlay", ainput=inputA, atype='area', binput=inputB,
      btype='area', operator='or', output=saida, overwrite=True, run_=False)
      uniao()
def cria_objTable(tab, path):
      from grass.pygrass.vector.table import Table, get_path
      import sqlite3
      objTabela = Table(name=tab, connection=sqlite3.connect(get_path(path)))
      return objTabela
def identifica_cat(tabela, grassdata_workspace, location, mapset):
      caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
      \\sqlite\\sqlite.db
      table = cria_objTable(tabela, caminho)
      for linha in table:
         if linha[1] == None and linha[2] == 1:
            cat = int(linha[0])
      return cat
def reclass(raster, saida, identi):
      import os
      os.mkdir("C:\\temporario")
      txt = open("C:\\temporario\\regras.txt", "w")
      txt.write(str(identi) + "  = 0 \n")
      txt.write("*  = NULL")
      txt.close()
      regras = "C:\\temporario\\regras.txt"
      from grass.pygrass.modules import Module
      reclassify = Module("r.reclass", input=raster, output=saida, rules=regras,
      overwrite=True, run_=False)
      reclassify()
      import shutil
      shutil.rmtree("C:\\temporario")
def itera_shape(vector):
      from grass.pygrass.vector import VectorTopo
      layer = VectorTopo(vector) layer.is_open()
      layer.exist()
      layer.mapset layer.open(mode='r')
      linha = layer.next()
      conta = 0
      try:
         while linha:
            conta += 1
            linha = layer.next()
      except:
         layer.close()
      return conta
def RasterToPolygon(raster, vector):
      from grass.pygrass.modules import Module
      raster2poly = Module("r.to.vect", input=raster, output=vector, type='area',
      column='value', overwrite=True, run_=False)
      raster2poly()
def create_random_points(vector, nr_pontos, saida):
```

```
    from grass.pygrass.modules import Module
    aleatorio = Module("v.random", output=saida, npoints=nr_pontos, restrict=vector,
    overwrite=True, run_=False)
    aleatorio()
def add_coluna(vector):
    from grass.pygrass.modules import Module
    coluna = Module("v.db.addcolumn", map=vector, columns='extract DOUBLE PRECISION',
    run_=False)
    coluna()
def extract_values(pontos, rst):
    from grass.pygrass.modules import Module
    extract = Module("v.what.rast", map=pontos, type="point", raster=rst,
    column="extract", run_=False)
    extract()
def create_table(vector):
    from grass.pygrass.modules import Module
    create = Module("v.db.addtable", map=vector, columns='extract DOUBLE PRECISION',
    run_=False)
    create()
def v_out_ogr(g_vect, vector):
    from grass.pygrass.modules import Module
    out_vector = Module("v.out.ogr", input=g_vect, type="point", output=vector,
    format="ESRI_Shapefile", overwrite=True, run_=False)
    out_vector()
def __main__GRASS(fenomeno, amostra_validacao, susceptibilidade, limites,
grassdata, location, mapset, resultados):
    v_in_ogr(fenomeno, "fenomeno")
    v_in_ogr(amostra_validacao, "p_validacao")
    buffert("fenomeno", "bf_fenomeno")
    v_in_ogr(limites, "lmt")
    union("bf_fenomeno","lmt" , "lmt_bf")
    cat = identifica_cat("lmt_bf_1", grassdata, location, mapset)
    v2rst("lmt_bf", "rst_lmt_bf")
    reclass("rst_lmt_bf", "reclassificado", cat)
    nr_pnt = itera_shape("p_validacao")
    RasterToPolygon("reclassificado", "v_erase")
    create_random_points("v_erase", nr_pnt, "ausencias")
    r_in_gdal(susceptibilidade, "rst_vi")
    l = ["p_validacao", "ausencias"]
    create_table("ausencias")
    for i in l:
       add_coluna(i)
       extract_values(i, "rst_vi")
       v_out_ogr(i, resultados + "\\" + i + ".shp")
```

Programa 33 - Bloco de código (em Python, v. 2.7.4) que gera as ausências e extrai os valores das células do *raster* de susceptibilidade que se intersectam com as presenças e ausências, guardando essa informação na tabela de atributos desse tema pontual – QGIS 2.10.1.

```
# Programa que adiciona as ausências à shapefile que representa a amostra de
validação (que corresponde a 30% das presenças). Tendo as presenças e ausências
no mesmo tema vectorial, o Programa também extrai, do raster que se pretende
validar, os valores das células que se intersectam com os pontos – QGIS
import os, shutil, processing
from qgis.core import QgsRasterLayer, QgsVectorLayer, QgsProviderRegistry,
QgsVectorFileWriter, QgsGeometry, QgsFeature, QgsPoint
from PyQt4.QtCore import QVariant
from osgeo import gdal, ogr
##MovVert= group
##ValInformativo= name
```

```
##AmostraTotal= vector
##AmostraValidacao= vector
##Susceptibilidade= raster
##Limites= vector
##saida= output vector
def cria_layer(objecto, nome, rst_ou_vect):
    if rst_ou_vect == True:
       rst_lyr = QgsRasterLayer(objecto, nome)
       return rst_lyr
    elif rst_ou_vect == False:
       v_lyr = QgsVectorLayer(objecto, nome, "ogr")
       return v_lyr
def itera_shape(shape):
    registo = QgsProviderRegistry.instance()
    provider = registo.provider("ogr", shape)
    conta = 0
    for feature in provider.getFeatures():
       conta += 1
    return conta
def merge(presencas, ausencias, saida):
    def criar_shp_aPartirDeOutra(entrada, destino):
       lyr = cria_layer(entrada, "indf", False)
       provider = lyr.dataProvider()
       writer = QgsVectorFileWriter(destino, provider.encoding(), provider.fields(),
       ogr.wkbPoint,provider.crs())
    criar_shp_aPartirDeOutra(presencas, saida)
    def cria_campo(shape, campo):
       lyr = cria_layer(shape, "indf2", False)
       vpr = lyr.dataProvider()
       vpr.addAttributes([QgsField(campo, QVariant.Int)])
    cria_campo(saida, "fenom")
    def extrair_pontos(shape):
       registo = QgsProviderRegistry.instance()
       provider = registo.provider("ogr", shape)
       lst_pontos = []
       for feature in provider.getFeatures():
          geom = feature.geometry()
          ponto = geom.asPoint()
          lst_pontos.append([ponto[0], ponto[1]])
       return lst_pontos
    pres = extrair_pontos(presencas)
    aus = extrair_pontos(ausencias)
    def adiciona_pontos(shape, lista, momento, campo):
       lyr = cria_layer(shape, "out", False)
       vpr = lyr.dataProvider()
       for ponto in lista:
          pnt = QgsGeometry.fromPoint(QgsPoint(ponto[0], ponto[1]))
          fields = vpr.fields()
          f = QgsFeature(fields)
          f.setGeometry(pnt)
          f[campo] = momento
          vpr.addFeatures([f])
          lyr.updateExtents()
    adiciona_pontos(saida, pres, 1, "fenom")
    adiciona_pontos(saida, aus, 0, "fenom")
def __main__QGIS(fenomeno, amostra_validacao, susceptibilidade, limites, final):
    w = "C:\\areatrab"
    os.mkdir(w)
    processing.runalg("qgis:fixeddistancebuffer", cria_layer(fenomeno, "fenomeno",
    False), 10, 5, True, w + "\\bf_mov.shp")
    processing.runalg("qgis:difference", cria_layer(limites, "lmt", False), cria_layer(w
    + "\\bf_mov.shp", "buffer", False), w + "\\lmt_erase.shp")
    nr_pnt = itera_shape(amostra_validacao)
```

```
    processing.runalg("qgis:randompointsinsidepolygonsfixed", cria_layer(w +
    "\\lmt_erase.shp", "erase", False), 0, nr_pnt, 10, w + "\\ausencias.shp")
    merge(amostra_validacao, w + "\\ausencias.shp", w + "\\validacao.shp")
    processing.runalg("saga:addgridvaluestopoints", w + "\\validacao.shp",
    susceptibilidade, 0, final)
__main__QGIS(AmostraTotal, AmostraValidacao, Susceptibilidade, Limites, saida)
```

Programa 34 - Bloco de código (em Python, v. 2.7.4) que gera as ausências e extrai os valores das células do *raster* de susceptibilidade que se intersectam com as presenças e ausências, guardando essa informação na tabela de atributos desse tema pontual – GDAL/OGR 1.11.1.

```
# Programa que adiciona as ausências à shapefile que representa a amostra de
validação (que corresponde a 30% das presenças). Tendo as presenças e ausências
no mesmo tema vectorial, o Programa também extrai, do raster que se pretende
validar, os valores das células que se intersectam com os pontos – GDAL/OGR
import os, shutil, random, numpy
from osgeo import gdal, ogr
def itera_shape(shapefile):
     driver = ogr.GetDriverByName("ESRI Shapefile")
     dataSource = driver.Open(shapefile, 0)
     layer = dataSource.GetLayer()
     conta = 0
     for feature in layer:
        conta += 1
     return conta
def CreateRandomPoint(minX, maxX, minY, maxY):
     x = minX + (random.random() * (maxX - minX))
     y = minY + (random.random() * (maxY - minY))
     return [x, y]
def extrair_presencas(shape):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    shp = driver.Open(shape)
    lyr = shp.GetLayer()
    lista = []
    for feat in lyr:
       geom = feat.GetGeometryRef()
       coord = str.split(str(geom), " ")
       x = float(coord[1][1:])
       y = float(coord[2][:-1])
       lista.append([x, y])
    return lista
def __main__GDAL(fenomeno, amostra_validacao, susceptibilidade, limites, saida):
    nr_pnt = itera_shape(amostra_validacao)
    driver = ogr.GetDriverByName("ESRI Shapefile")
    shp = driver.Open(limites)
    layer = shp.GetLayer()
    min_x, max_x, min_y, max_y = layer.GetExtent()
    driver = ogr.GetDriverByName("ESRI Shapefile")
    mov = driver.Open(fenomeno)
    lyr_mov = mov.GetLayer()
    ausencias = []
    for i in range(nr_pnt):
       igual = -1
       while igual != 0:
          pnt_aleatorio = CreateRandomPoint(min_x, max_x, min_y, max_y)
          igual = 0
          for feature in lyr_mov:
             geom = feature.GetGeometryRef()
             coord = str.split(str(geom), " ")
             x = float(coord[1][1:])
```

```
            y = float(coord[2][:-1])
            distancia = abs(((x - pnt_aleatorio[0])**2 + (y - pnt_aleatorio[1])**2)**0.5)
            if distancia < 10:
               igual += 1
            if igual == 0:
               ausencias.append(pnt_aleatorio)
    presencas = extrair_presencas(amostra_validacao)
    driver = ogr.GetDriverByName("ESRI Shapefile")
    output = driver.CreateDataSource(saida)
    out_layer = output.CreateLayer(saida, geom_type=ogr.wkbPoint)
    campo = ogr.FieldDefn("fenom", ogr.OFTInteger)
    out_layer.CreateField(campo)
    featureDefn = out_layer.GetLayerDefn()
    for coord in presencas:
       feature = ogr.Feature(featureDefn)
       ponto = ogr.Geometry(ogr.wkbPoint)
       ponto.AddPoint(coord[0], coord[1])
       feature.SetGeometry(ponto)
       feature.SetField("fenom", 1)
       out_layer.CreateFeature(feature)
    for coord in ausencias:
       feature = ogr.Feature(featureDefn)
       ponto = ogr.Geometry(ogr.wkbPoint)
       ponto.AddPoint(coord[0], coord[1])
       feature.SetGeometry(ponto)
       feature.SetField("fenom", 0)
       out_layer.CreateFeature(feature)
    output.Destroy()
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(saida, 1)
    layer = dataSource.GetLayer()
    campo = ogr.FieldDefn("susc", ogr.OFTReal)
    layer.CreateField(campo)
    img = gdal.Open(susceptibilidade)
    geo_transform = img.GetGeoTransform()
    band = img.GetRasterBand(1)
    for feat in layer:
       geom = feat.GetGeometryRef()
       mx, my = geom.GetX(), geom.GetY()
       px = int((mx - geo_transform[0]) / geo_transform[1])
       py = int((my - geo_transform[3]) / geo_transform[5])
       val_pix = band.ReadAsArray(px, py, 1, 1)
       feat.SetField("susc", float(val_pix[0][0]))
       layer.SetFeature(feat)
    dataSource.Destroy()
```

Programa 35 - Alterações ao Programa 20, tendo em vista a optimização da execução da tarefa em estudo.

```
def db_in_ogr(tabela, nome):
    from grass.pygrass.modules import Module
    adiciona = Module("db.in.ogr", input=tabela, output=nome, run_=False)
    adiciona()
def conta_celula(tabela, grassdata_workspace, location, mapset):
    def cria_objTable(tab, path):
       from grass.pygrass.vector.table import Table, get_path
       import sqlite3
       objTabela = Table(name=tab, connection=sqlite3.connect(get_path(path)))
       return objTabela
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +  \\sqlite\\sqlite.db
    tabela = cria_objTable(tabela, caminho)
```

```
    conta = 0
    for linha in tabela:
       conta += int(linha[1])
    return conta
def nr_pixel_classe(tabela, grassdata_workspace, location, mapset):
    def cria_objTable(tabela, path):
       from grass.pygrass.vector.table import Table, get_path
       import sqlite3
       objTabela = Table(name=tabela, connection=sqlite3.connect(get_path(path)))
       return objTabela
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
    \\sqlite\\sqlite.db
    tabela = cria_objTable(tabela, caminho)
    dic = {}
    for linha in tabela:
       temp = {int(linha[0]):int(linha[1])}
       dic.update(temp)
    return dic
```

Programa 36 - Alterações aos Programas 21 e 22, tendo em vista a optimização da execução da tarefa em estudo.

```
def conta_nrpixeis(tabela_raster):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(tabela_raster, 0)
    layer = dataSource.GetLayer()
    conta = 0
    for linha in layer:
       count = int(linha.GetField('Count'))
       conta += count

    return conta
def nr_pixel_classe(table_raster):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    dataSource = driver.Open(table_raster, 0)
    layer = dataSource.GetLayer()
    dic = {}
    for linha in layer:
       temp = {int(linha.GetField('Value')):int(linha.GetField('Count'))}
       dic.update(temp)
    return dic
```

Programa 37 - Alterações ao Programa 19 que visam obter um indicador de permita comparar as soluções em confronto quanto à demora na leitura (célula a célula) dos factores condicionantes.

```
def calcula_unitArea(raster, tamanho_celula, nr_movimentos):
    array = arcpy.RasterToNumPyArray(raster)
    nr_pixeis = 0
    for i in array:
       for e in i:
          if e < 0 or e > 20:
             continue
          else:
             nr_pixeis += 1
    unit_area = ((((tamanho_celula * tamanho_celula) * float(nr_pixeis)) / 1000000.0) /
    nr_movimentos) / 40.0
    return unit_area
def nr_pixel_classe(raster):
```

```
    import numpy
    array = arcpy.RasterToNumPyArray(raster)
    values = numpy.unique(array)
    lista = []
    for i in values:
       if i < 0 or i > 20:
          continue
       else: lista.append(i)
    dic = {}
    for i in lista:
       conta = 0
       for e in array:
          temp = list(e)
          elementos = temp.count(i)
          conta += elementos
          dic_temp = {i:conta}
          dic.update(dic_temp)
    return dic
```

Programa 38 - *Script* (em Python, v. 2.7.3) que constrói uma *feature class* (linear ou poligonal) a partir de um ficheiro texto, onde se refere o número de identificação da entidade geométrica, o número de partes e as coordenadas de todos os vértices das linhas ou polígonos.

```
import arcpy
texto = arcpy.GetParameterAsText(0)
workspace = arcpy.GetParameterAsText(1)
fc = arcpy.GetParameterAsText(2)
primitiva_geometrica = arcpy.GetParameterAsText(3)
prj = arcpy.GetParameterAsText(4)
texto = open(texto, "r")
arcpy.env.workspace = workspace
arcpy.CreateFeatureclass_management(workspace, fc, primitiva_geometrica, "", "", "", prj)
cursor = arcpy.InsertCursor(fc)
vector_mae = arcpy.CreateObject("Array")
vector_1 = arcpy.CreateObject("Array")
ponto = arcpy.CreateObject("Point")
ler_texto = texto.readlines()
tempID = -1
tempParte = -1
for linha in ler_texto:
    valores = str.split(linha, " ")
    linhaID = valores[0]
    parteID = valores[1]
    if tempID == -1:
       tempID = linhaID
    if tempID == linhaID:
       if tempParte == -1:
          tempParte = parteID
       if tempParte == parteID:
          ponto.X = valores[2]
          ponto.Y = valores[3]
          vector_1.add(ponto)
       else:
          vector_mae.add(vector_1)
          vector_2 = 0
          vector_2 = arcpy.CreateObject("Array")
          ponto.X = valores[2]
          ponto.Y = valores[3]
          vector_2.add(ponto)
          vector_1 = vector_2
```

```
            tempParte = parteID
        else:
            vector_mae.add(vector_1)
            inserir = cursor.newRow()
            inserir.Shape = vector_mae
            cursor.insertRow(inserir)
            vector_mae_2 = 0
            vector_mae_2 = arcpy.CreateObject("Array")
            vector_1 = 0
            vector_1 = arcpy.CreateObject("Array")
            ponto.X = valores[2]
            ponto.Y = valores[3]
            vector_1.add(ponto)
            vector_mae = vector_mae_2
            tempID = linhaID
            tempParte = 0
vector_mae.add(vector_1)
inserir = cursor.newRow()
inserir.Shape = vector_mae
cursor.insertRow(inserir)
texto.close()
del vector_mae, vector_1, ponto, vector_mae_2
```

Programa 39 - Bloco de Código (em Python, v. 2.7.5) estabelecido para o QGIS 2.10.1 que produz uma cópia de uma *shapefile*, garantido a permanência de espaços vazios dentro dos polígonos.

```
def criar_shp_aPartirDeOutra(entrada, destino):
    from qgis.core import QgsVectorFileWriter
    from osgeo import ogr
    lyr = cria_layer(entrada, "indf")
    provider = lyr.dataProvider()
    writer = QgsVectorFileWriter(destino, provider.encoding(), provider.fields(),
    ogr.wkbPolygon,provider.crs())
def extrair_coord(shapefile, tempo):
    registo = QgsProviderRegistry.instance()
    provider = registo.provider("ogr", shapefile)
    i_poligonos = {}
    i_ilhas = {}
    id_feature = 0
    for feature in provider.getFeatures():
        geometria = feature.geometry()
        ilhas = []
        poligono = 1
        for polygon in geometria.asPolygon():
            if poligono == 1:
                i_poligonos.update({id_feature:polygon})
            else:
                ilhas.append(polygon)
            poligono += 1
        i_ilhas.update({id_feature:ilhas})
        id_feature += 1
    if tempo == 2:
        return i_poligonos
    elif tempo == 3:
        return i_ilhas
def adiciona_polygon_shp(lyr, objecto, dic_poligonos, dic_ilhas):
    for chave in dic_poligonos.keys():
        points = []
        for ponto in dic_poligonos[chave]:
            points.append(QgsPoint(ponto[0], ponto[1]))
        fields = objecto.fields()
```

```
        f = QgsFeature(fields)
        holes = []
        for ilha in dic_ilhas[chave]:
           points2 = []
           for ponto in ilha:
              points2.append(QgsPoint(ponto[0], ponto[1]))
           holes.append(points2)
        lst = []
        lst.append(points)
        for i in holes:
           lst.append(i)
        poly = QgsGeometry.fromPolygon(lst)
        f.setGeometry(poly)
        objecto.addFeatures([f])
        lyr.updateExtents()
def cria_layer(objecto, nome):
     from qgis.core import QgsVectorLayer
     v_lyr = QgsVectorLayer(objecto, nome, "ogr")
     return v_lyr
def run(entrada, saida):
     criar_shp_aPartirDeOutra(entrada, saida)
     poligonos = extrair_coord(entrada, 2)
     ilhas = extrair_coord(entrada, 3)
     vectorLyr = cria_layer(saida, "poligonos")
     vpr = vectorLyr.dataProvider()
     adiciona_polygon_shp(vectorLyr, vpr, poligonos, ilhas)
```

Programa 40 - *Script* (em Python, v. 2.7.3) que implementa, no ArcGIS 10.2, o algoritmo que visa a obtenção de uma grelha regular cujas unidades matriciais envergam um indicador de impedância (em tempo), que um indivíduo terá de ultrapassar para a atravessar – criação de uma superfície de custo.

```
import arcpy
import unicodedata
mdt = arcpy.GetParameterAsText(0)
cos = arcpy.GetParameterAsText(1)
barreiras = arcpy.GetParameterAsText(2)
vias = arcpy.GetParameterAsText(3)
workspace = arcpy.GetParameterAsText(4)
custo = arcpy.GetParameterAsText(5)
arcpy.env.overwriteOutput = True
arcpy.env.workspace = workspace
declv = "declv.img"
arcpy.gp.Slope_sa(mdt, declv, "PERCENT_RISE")
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
rclss_declv = "reclssDeclv.img"
arcpy.gp.Reclassify_sa(declv, "Value", "0 10 1; 10 30 2; 30 50 3; 50 70 4; 70 100 5; 100
500 6", rclss_declv, "DATA")
arcpy.env.extent = tempEnvironment0
arcpy.AddField_management(cos, "leg", "TEXT", "50", "", "", "", "NULLABLE",
"NON_REQUIRED", "")
up_cursor = arcpy.UpdateCursor(cos)
for linha in up_cursor:
     classe = linha.getValue("COSN2")
     if classe[0] == "1":
        linha.setValue("leg", "Urbano")
        up_cursor.updateRow(linha)
     if classe[0] == "2":
        linha.setValue("leg", "Agricola")
        up_cursor.updateRow(linha)
     if classe == "3.1":
```

```
         linha.setValue("leg", "FlorestasMatosIncultos")
         up_cursor.updateRow(linha)
      if classe == "3.2":
         linha.setValue("leg", "FlorestasMatosIncultos")
         up_cursor.updateRow(linha)
      if classe =="3.3":
         linha.setValue("leg", "TerrenosSemCobertura")
         up_cursor.updateRow(linha)
      if classe[0] == "4":
         linha.setValue("leg", "Zonas Húmidas")
         up_cursor.updateRow(linha)
      if classe[0] == "5":
      linha.setValue("leg", "Corpos Água")
      up_cursor.updateRow(linha)
str_union = barreiras + ";" + cos
barr_cos = "barr_cos.shp"
arcpy.Union_analysis(str_union, barr_cos, "ALL", "", "GAPS")
up_cursor = arcpy.UpdateCursor(barr_cos)
for i in up_cursor:
      fid_barr = i.getValue("FID_barrei")
      if fid_barr == 0:
         i.setValue("leg", "barreira")
         up_cursor.updateRow(i)
del up_cursor, i, fid_barr
rst_barrCos = "rst_cos.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.PolygonToRaster_conversion(barr_cos, "leg",rst_barrCos, "CELL_CENTER", "NONE",
"10")
arcpy.env.extent = tempEnvironment0
rclss_BARR_COS = "rclssBarrCos.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.gp.Reclassify_sa(rst_barrCos, "leg", "'barreira' NODATA;'Urbano' 1; 'Agricola' 2;
'FlorestasMatosIncultos' 3;'TerrenosSemCobertura' 4; 'Zonas Húmidas' 5;'Corpos Água'
NODATA", rclss_BARR_COS, "DATA")
arcpy.env.extent = tempEnvironment0
rst_vias = "rst_vias.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.PolylineToRaster_conversion(vias, "KPH", rst_vias, "MAXIMUM_LENGTH", "NONE", "10")
arcpy.env.extent = tempEnvironment0
cursor = arcpy.SearchCursor(rst_vias, "", "", "", "")
linha = cursor.next()
logos = 0
while linha:
      velocidade = linha.getValue("Value")
      if logos == 0:
         velocidades = str(velocidade) + " " + str(velocidade)
      else:
         velocidades = velocidades + ";" + str(velocidade) + " " + str(velocidade)
      logos += 1
      linha = cursor.next()
del logos, cursor, velocidade, linha
velocidades2 = velocidades + ";NODATA 0"
rclss_vias = "rclss_vias.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.gp.Reclassify_sa(rst_vias, "Value", velocidades2, rclss_vias, "DATA")
arcpy.env.extent = tempEnvironment0
in_mosaic = workspace + "\\" + rclss_BARR_COS + ";" + workspace + "\\" + rclss_vias
barr_cos_vias = "BCOS_VIAS.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
```

```
arcpy.MosaicToNewRaster_management(in_mosaic, workspace, barr_cos_vias, "",
"8_BIT_UNSIGNED", "10", "1", "MAXIMUM", "FIRST")
arcpy.env.extent = tempEnvironment0
velocidades = "0 NODATA;" + velocidades
barr_cos_vias_ND = "BCOSVIASND.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.gp.Reclassify_sa(barr_cos_vias, "Value", velocidades, barr_cos_vias_ND, "DATA")
arcpy.env.extent = tempEnvironment0
combine = workspace + "\\" + barr_cos_vias_ND + ";" + workspace + "\\" + rclss_declv
rst_combine = "combine.img"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.gp.Combine_sa(combine, rst_combine)
arcpy.env.extent = tempEnvironment0
logos = 0
for i in range(3):
      if logos == 0:
         campo = "Pdeclv"
      else:
         if logos == 1:
            campo = "PCOS"
         else:
            campo = "preCust"
      arcpy.AddField_management(rst_combine, campo, "FLOAT", "6", "3", "", "","NULLABLE",
      "NON_REQUIRED", "")
      logos += 1
lista_campos = [f.name for f in arcpy.ListFields(rst_combine)]
for i in range(len(lista_campos)):
      lista_campos[i] = unicodedata.normalize('NFKD', lista_campos[i]).encode('ascii',
      'ignore')
up_cursor = arcpy.UpdateCursor(rst_combine)
for linha in up_cursor:
      declv = linha.getValue(lista_campos[4])
      if declv == 1:
         linha.setValue("Pdeclv", 1)
         up_cursor.updateRow(linha)
      if declv == 2:
         fora_dentro = linha.getValue(lista_campos[3])
         if fora_dentro >= 6:
            linha.setValue("Pdeclv", 1)
            up_cursor.updateRow(linha)
         else:
            linha.setValue("Pdeclv", 1.5)
            up_cursor.updateRow(linha)
      if declv == 3:
         fora_dentro = linha.getValue(lista_campos[3])
         if fora_dentro >= 6:
            linha.setValue("Pdeclv", 1.5)
            up_cursor.updateRow(linha)
         else:
            linha.setValue("Pdeclv", 2)
            up_cursor.updateRow(linha)
      if declv == 4:
         fora_dentro = linha.getValue(lista_campos[3])
         if fora_dentro >= 6:
            linha.setValue("Pdeclv", 1.5)
            up_cursor.updateRow(linha)
         else:
            linha.setValue("Pdeclv", 3)
            up_cursor.updateRow(linha)
      if declv == 5:
         fora_dentro = linha.getValue(lista_campos[3])
         if fora_dentro >= 6:
```

```
            linha.setValue("Pdeclv", 2)
            up_cursor.updateRow(linha)
         else:
            linha.setValue("Pdeclv", 4)
            up_cursor.updateRow(linha)
      if declv == 6:
         fora_dentro = linha.getValue(lista_campos[3])
         if fora_dentro >= 6:
            linha.setValue("Pdeclv", 2)
            up_cursor.updateRow(linha)
         else:
            linha.setValue("Pdeclv", 5)
            up_cursor.updateRow(linha)
del up_cursor, linha, declv, fora_dentro
up_cursor = arcpy.UpdateCursor(rst_combine)
for linha in up_cursor:
      cos_vel = linha.getValue(lista_campos[3])
      if cos_vel >= 6:
         tempo = (3600.0 * 10) / (cos_vel * 1000.0)
         linha.setValue("PCOS", tempo)
         up_cursor.updateRow(linha)
      else:
         if cos_vel == 1:
            linha.setValue("PCOS", 9)
            up_cursor.updateRow(linha)
         if cos_vel == 2:
            linha.setValue("PCOS", 18)
            up_cursor.updateRow(linha)
         if cos_vel == 3:
            linha.setValue("PCOS", 15)
            up_cursor.updateRow(linha)
         if cos_vel == 4:
            linha.setValue("PCOS", 12)
            up_cursor.updateRow(linha)
         if cos_vel == 5:
         linha.setValue("PCOS", 23)
         up_cursor.updateRow(linha)
arcpy.CalculateField_management(rst_combine, "preCust", "[Pdeclv] * [PCOS]", "VB", "")
entrada = workspace + "\\" + rst_combine + " preCust 1"
tempEnvironment0 = arcpy.env.extent
arcpy.env.extent = mdt
arcpy.gp.WeightedSum_sa(entrada, custo)
arcpy.env.extent = tempEnvironment0
```

Programa 41 - *Script* (em Python, v. 2.7.4) que implementa, no GRASS GIS 7.0.0, o algoritmo que visa a obtenção de uma grelha regular cujas unidades matriciais envergam um indicador de impedância (em tempo), que um indivíduo terá de ultrapassar para a atravessar – criação de uma superfície de custo.

```
def r_in_gdal(raster, grass_raster):
    from grass.pygrass.modules import Module
    adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
    overwrite=True, run_=False)
    adiciona()
def declives(mdt, declv):
    from grass.pygrass.modules import Module
    slope = Module("r.slope.aspect", elevation=mdt, slope=declv, format='percent',
    overwrite=True, precision='FCELL', run_=False)
    slope()
def reclass(raster, saida, momento, dic):
    import os
```

```
    os.mkdir("C:\\temporario")
    txt = open("C:\\temporario\\regras.txt", "w")
    if momento == 1:
        txt.write("0 thru 10  = 1\n")
        txt.write("10 thru 30  = 2\n")
        txt.write("30 thru 50  = 3\n")
        txt.write("50 thru 70  = 4\n")
        txt.write("70 thru 100  = 5\n")
        txt.write("100 thru 500  = 6")
        txt.close()
    if momento == 2:
        txt.write("1  = 1\n")
        txt.write("2  = 2\n")
        txt.write("3  = 3\n")
        txt.write("4  = 4\n")
        txt.write("5  = 5\n")
        txt.write("6  = NULL\n")
        txt.write("7  = NULL")
        txt.close()
    if momento == dic:
       for i in momento.keys():
          custo_pormilhao = momento[i] * 10000000.0
          txt.write(str(i) + "   = " + str(custo_pormilhao) + "\n")
       txt.close()
    regras = 'C:\\temporario\\regras.txt'
    from grass.pygrass.modules import Module
    reclassify = Module("r.reclass", input=raster, output=saida, rules=regras,
    overwrite=True, run_=False)
    reclassify()
    import shutil
    shutil.rmtree("C:\\temporario")
def v_in_ogr(shapefile, grass_vector):
    from grass.pygrass.modules import Module
    adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
    overwrite=True, run_=False)
    adiciona()
def edita_tabela(grass_vector, grassdata_workspace, location, mapset):
    def cria_coluna(vector, path):
       from grass.pygrass.vector.table import Columns, get_path
       import sqlite3
       cols = Columns(grass_vector, sqlite3.connect(get_path(caminho)))
       cols.add(['legen'], ['INT'])
    def cria_objTable(vector, path):
       from grass.pygrass.vector.table import Table, get_path
       import sqlite3
       objTabela = Table(name=vector, connection=sqlite3.connect(get_path(path)))
       return objTabela
    caminho = grassdata_workspace + "\\" + location + "\\" + mapset +
    "\\sqlite\\sqlite.db"
    cria_coluna(grass_vector, caminho)
    tabela = cria_objTable(grass_vector, caminho)
    import sqlite3
    table_name = grass_vector
    id_col = 'cat'
    col_name = 'legen'
    conn = sqlite3.connect(caminho)
    c = conn.cursor()
    conta = 1
    for linha in tabela:
       if linha[1][0] == u'1':
          c.execute("UPDATE {tn} SET {cn}=(1) WHERE {idf}={val}".format(tn=table_name,
          idf=id_col, cn=col_name, val=conta))
       if linha[1][0] == u'2':
```

```
            c.execute("UPDATE {tn} SET {cn}=(2) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         if linha[1] == u'3.1':
            c.execute("UPDATE {tn} SET {cn}=(3) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         if linha[1] == u'3.2':
            c.execute("UPDATE {tn} SET {cn}=(3) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         if linha[1] == u'3.3':
            c.execute("UPDATE {tn} SET {cn}=(4) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         if linha[1][0] == u'4':
            c.execute("UPDATE {tn} SET {cn}=(5) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         if linha[1][0] == u'5':
            c.execute("UPDATE {tn} SET {cn}=(6) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         conta += 1
      conn.commit()
      conn.close()
   def union(inputA, inputB, saida):
      from grass.pygrass.modules import Module
      uniao = Module("v.overlay", ainput=inputA, atype='area', binput=inputB,
      btype='area', operator='or', output=saida, overwrite=True, run_=False)
      uniao()
   def edita_union(entrada, workspace, location, mapset):
      def cria_objTable(vector, path):
         from grass.pygrass.vector.table import Table, get_path
         import sqlite3
         objTabela = Table(name=vector, connection=sqlite3.connect(get_path(path)))
         return objTabela
      import sqlite3
      caminho = workspace + "\\" + location + "\\" + mapset + "\\sqlite\\sqlite.db"
      tabela = cria_objTable(entrada, caminho)
      table_name = entrada
      id_col = 'cat'
      col_name = 'b_legen'
      conn = sqlite3.connect(caminho)
      c = conn.cursor()
      conta = 1
      for linha in tabela:
         if linha[1] == 1:
            c.execute("UPDATE {tn} SET {cn}=(7) WHERE {idf}={val}".format(tn=table_name,
            idf=id_col, cn=col_name, val=conta))
         conta += 1
      conn.commit()
      conn.close()
   def v_to_raster(vector, raster, col, momento):
      from grass.pygrass.modules import Module
      if momento == 1:
         vec_to_rst = Module("v.to.rast", input=vector, type='area', output=raster,
         use='attr', attribute_column=col, overwrite=True, run_=False)
      if momento == 2:
         vec_to_rst = Module("v.to.rast", input= vector, type='line', output=raster,
         use='attr', attribute_column=col, overwrite=True, run_=False)
      vec_to_rst()
   def mosaic_raster(inputA, inputB, saida):
      from grass.pygrass.modules import Module
      mosaic = Module("r.patch", input=[inputA, inputB], output=saida, overwrite=True,
      run_=False)
      mosaic()
   def combine_raster(inputA, inputB, saida):
      from grass.pygrass.modules import Module
```

```
    combine = Module("r.cross", input=[inputA, inputB], output=saida, flags='z',
    overwrite=True, run_=False)
    combine()
def relatorio_rst(mapa, saida):
    from grass.pygrass.modules import Module
    report = Module("r.report", map=mapa, flags='h', output=saida, run_=False)
    report()
def calcula_custo(dic):
    dc = {}
    for chave in dic.keys():
       class_decliv = dic[chave][0]
       if class_decliv == '1':
          peso = 1
       if class_decliv == '2':
          cos_vias = dic[chave][1]
          if int(cos_vias) >= 6:
             peso =1
          else:
             peso =1.5
       if class_decliv == '3':
          cos_vias = dic[chave][1]
          if int(cos_vias) >= 6:
             peso = 1.5
          else:
             peso = 2
       if class_decliv == '4':
          cos_vias = dic[chave][1]
          if int(cos_vias) >= 6:
             peso =1.5
          else:
             peso = 3
       if class_decliv == '5':
          cos_vias = dic[chave][1]
          if int(cos_vias) >= 6:
             peso = 2
          else:
             peso = 4
       if class_decliv == '6':
          cos_vias = dic[chave][1]
          if int(cos_vias) >= 6:
             peso = 2
          else:
             peso = 5
       temp = {chave:[peso, dic[chave][1]]}
       dc.update(temp)
    novo = {}
    for chave in dc.keys():
       cos_vel = int(dc[chave][1])
       if cos_vel >= 6:
          tempo = (3600.0* 10) / (cos_vel * 1000.0)
       else:
          if cos_vel == 1:
             tempo = 9.0
          if cos_vel == 2:
             tempo = 18.0
          if cos_vel == 3:
             tempo = 15.0
          if cos_vel == 4:
             tempo = 12.0
          if cos_vel == 5:
             tempo = 23.0
       temp = {chave:[dc[chave][0], tempo]}
       novo.update(temp)
    dic_cst = {}
    for chave in novo.keys():
       custo = novo[chave][0] * novo[chave][1]
```

```
        temp = {chave:custo}
        dic_cst.update(temp)
    return dic_cst
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
def sup_custo(mdt, cos, barreiras, vias, sup_custo, g_workspace, g_location,
g_mapset):
    r_in_gdal(mdt, 'mdt')
    declives('mdt', 'declv')
    reclass('declv', 'declv.recl', 1)
    v_in_ogr(cos, 'cos')
    edita_tabela('cos', g_workspace, g_location, g_mapset)
    v_in_ogr(barreiras, 'barr')
    union('barr', 'cos', 'barrcos')
    edita_union('barrcos_1', g_workspace, g_location, g_mapset)
    v_to_raster('barrcos', 'rst_barrcos', 'b_legen', 1)
    reclass('rst_barrcos', 'recls_barrcos', 2)
    v_in_ogr(vias, 'vias')
    v_to_raster('vias', 'rst_vias', 'KPH', 2)
    mosaic_raster('rst_vias', 'recls_barrcos', 'vias_barrcos')
    combine_raster('declv.recl', 'vias_barrcos', 'combine')
    import os
    os.mkdir("C:\\temporario")
    relatorio = "C:\\temporario\\relatorio.txt"
    relatorio_rst('combine', relatorio)
    texto = open(relatorio, "r")
    conta = 0
    dic = {}
    for linha in texto.readlines():
       try:
          if conta == 4:
             temp = {0:['1', '1']}
             dic.update(temp)
          if conta >= 5:
             parte_linha = str.split(linha, "|")
             categorias = str.split(parte_linha[2], "; ")
             categoria1 = str.split(categorias[0], " ")
             categoria2 = str.split(categorias[1], " ")
             temp ={int(parte_linha[1]):[categoria1[1], categoria2[1]]}
             dic.update(temp)
          conta += 1
       except:
          break
    texto.close()
    del relatorio, texto, conta, temp, parte_linha, categorias, categoria1, categoria2
    import shutil
    shutil.rmtree("C:\\temporario")
    dic_custo = calcula_custo(dic)
    reclass('combine', 'sup_custo', dic_custo, dic_custo)
    r_out_gdal('sup_custo', sup_custo)
```

Programa 42 - *Script* (em Python, v. 2.7.3) que executa a ferramenta do ArcGIS 10.2 (*Cost Distance*) que permite calcular o custo acumulado entre cada célula e um (ou vários) ponto de origem.

```
import arcpy
sup_custo = arcpy.GetParameterAsText(0)
infra_estrutura = arcpy.GetParameterAsText(1)
```

```
rst_distance = arcpy.GetParameterAsText(2)
arcpy.env.overwriteOutput = True
def cria_layer(objecto, nome_layer, rst_ou_vect):
    if rst_ou_vect == True:
       lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
       return lyr
    else:
       lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
       return lyr
def cst_distance(superficie_cst, destino, saida):
    lyr_sc = cria_layer(superficie_cst, "sup_cst", True)
    lyr_dst = cria_layer(destino, "infra", False)
    tempEnvironment0 = arcpy.env.extent
    arcpy.env.extent = superficie_cst
    arcpy.gp.CostDistance_sa(lyr_dst, lyr_sc, saida, "", "")
    arcpy.env.extent = tempEnvironment0
cst_distance(sup_custo, infra_estrutura, rst_distance)
```

Programa 43 - Bloco de código (em Python, v. 2.7.4) que executa a ferramenta do GRASS GIS 7.0.0 (*r.cost)* que permite calcular o custo acumulado entre cada célula e um (ou vários) ponto de origem.

```
def v_in_ogr(shapefile, grass_vector):
    from grass.pygrass.modules import Module
    adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
    overwrite=True, run_=False)
    adiciona()
def r_in_gdal(raster, grass_raster):
    from grass.pygrass.modules import Module
    adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
    overwrite=True, run_=False)
    adiciona()
def r_cost(custo, orig, saida):
    from grass.pygrass.modules import Module
    acmu_cst = Module("r.cost", input=custo, output=saida, start_points=orig,
    run_=False)
    acmu_cst()
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
def cst_distance(sup_custo, origem, rst_cst):
    r_in_gdal(sup_custo, "sup_custo")
    v_in_ogr(origem, "origem")
    r_cost("sup_custo", "origem", "isocronas")
    r_out_gdal("isocronas", rst_cst)
```

Programa 44 - Algoritmo (em Python, v. 2.7.3) que se propõe a gerar um cartograma de isócronas que representa o tempo (em cada célula) que alguém, caso solicitasse ajuda a partir desse local, teria de esperar até ser assistido e transportado para a unidade de saúde mais próxima, e que quantifica o número de indivíduos que estão a uma determinada distância dos equipamentos de prestação de socorro e da unidade saúde mais próxima – ArcGIS 10.2

```
import arcpy, os, shutil, xlwt
bombeiros = arcpy.GetParameterAsText(0)
cst_saude = arcpy.GetParameterAsText(1)
```

```
sup_cst = arcpy.GetParameterAsText(2)
pop_grid = arcpy.GetParameterAsText(3)
nr_pop = arcpy.GetParameterAsText(4)
lmt = arcpy.GetParameterAsText(5)
cst_dist_reclss = arcpy.GetParameterAsText(6)
relatorio = arcpy.GetParameterAsText(7)
teste = arcpy.GetParameterAsText(8)
def cria_layer(objecto, nome_layer, rst_ou_vect):
     if rst_ou_vect == True:
        lyr = arcpy.MakeRasterLayer_management(objecto, nome_layer, "", "", "1")
        return lyr
     else:
        lyr = arcpy.MakeFeatureLayer_management(objecto, nome_layer, "", "", "")
        return lyr
def cst_distance(superficie_cst, destino, saida):
     lyr_sc = cria_layer(superficie_cst, "sup_cst", True)
     lyr_dst = cria_layer(destino, "infra", False)
     tempEnvironment0 = arcpy.env.extent
     arcpy.env.extent = superficie_cst
     arcpy.gp.CostDistance_sa(lyr_dst, lyr_sc, saida, "", "")
     arcpy.env.extent = tempEnvironment0
def SecX10ToMinutes(entrada, saida, template):
     lyr = cria_layer(entrada, "entrada", True)
     expressao = "(" + "\"entrada\"" + "/ 10.0" + ") / 60.0"
     tempEnvironment0 = arcpy.env.extent
     arcpy.env.extent = template
     arcpy.gp.RasterCalculator_sa(expressao, saida)
     arcpy.env.extent = tempEnvironment0
def soma_raster(rst_in1, rst_in2, rst_out, template):
     cria_layer(rst_in1, "rst_in_um", True)
     cria_layer(rst_in2, "rst_in_dois", True)
     expressao = "\"rst_in_um\" + \"rst_in_dois\""
     tempEnvironment0 = arcpy.env.extent
     arcpy.env.extent = template
     arcpy.gp.RasterCalculator_sa(expressao, rst_out)
     arcpy.env.extent = tempEnvironment0
def cria_campo(raster, opcao):
     lyr = cria_layer(raster, "reclass", True)
     arcpy.AddField_management(lyr, "classe", "TEXT", "", "", "50", "", "NULLABLE",
     "NON_REQUIRED", "")
     up = arcpy.UpdateCursor(lyr)
     if opcao == "1":
        for u in up:
           id_classe = int(u.getValue("Value"))
           if id_classe == 1:
              u.setValue("classe", "entre0e15min"
              up.updateRow(u)
           elif id_classe == 2:
              u.setValue("classe", "entre15e30min"
              up.updateRow(u)
           elif id_classe == 3
              u.setValue("classe", "entre30e45min")
              up.updateRow(u)
           elif id_classe == 4:
              u.setValue("classe", "entre45e60min")
              up.updateRow(u)
           elif id_classe == 5:
              u.setValue("classe", "entre60e90min")
              up.updateRow(u)
           elif id_classe == 6:
              u.setValue("classe", "entre90e120min")
              up.updateRow(u)
           elif id_classe == 7:
              u.setValue("classe", "maior120")
```

```
            up.updateRow(u)
      elif opcao == "2":
         for u in up:
            id_classe = int(u.getValue("Value"))
            if id_classe == 1:
               u.setValue("classe", "entre60e90min")
               up.updateRow(u)
            elif id_classe == 2:
               u.setValue("classe", "entre90e120min")
               up.updateRow(u)
            elif id_classe == 3:
               u.setValue("classe", "entre120e150min")
               up.updateRow(u)
            elif id_classe == 4:
               u.setValue("classe", "entre150e180min")
               up.updateRow(u)
            elif id_classe == 5:
               u.setValue("classe", "entre180e240min")
               up.updateRow(u)
            elif id_classe == 6:
               u.setValue("classe", "maior240")
               up.updateRow(u)
def RasterToPolygon(raster, campo, limitess, workspace):
      arcpy.env.workspace = workspace
      lyr = arcpy.MakeRasterLayer_management(raster, "rst", "", "", "1")
      cs = arcpy.SearchCursor(lyr)
      linha = cs.next()
      logos = 0
      dic = {}
      while linha:
         classe = linha.getValue(campo)
         temp = {logos:classe}
         dic.update(temp)
         logos += 1
         linha = cs.next()
      del cs, linha, logos
      logos = 0
      for chave in dic.keys():
         if logos == 0:
            string = str(dic[chave]) + " " + str(chave)
         else:
            string = string + ";" + str(dic[chave]) + " " + str(chave)
         logos += 1
      string = string + "; NODATA 77"
      dic.update({77: 'nodata'})
      arcpy.gp.Reclassify_sa(lyr, campo, string, "iso_reclss.img", "DATA")
      del lyr
      lyr = arcpy.MakeRasterLayer_management(workspace + "\\iso_reclss.img","iso_reclss",
      "", "", "1")
      arcpy.RasterToPolygon_conversion(lyr, "v_isocronas_temp.shp", "NO_SIMPLIFY","Value")
      input_lyr = arcpy.MakeFeatureLayer_management(workspace +"\\v_isocronas_temp.shp",
      "input_lyr", "", "", "")
      clip_lyr = arcpy.MakeFeatureLayer_management(limitess, "lmt", "", "", "")
      arcpy.Clip_analysis(input_lyr, clip_lyr, "v_isocronas.shp")
      arcpy.Dissolve_management(workspace + "\\v_isocronas.shp", "d_isocronas.shp",
      "GRIDCODE", "", "MULTI_PART", "")
      return dic
def conta_pop(feature_pop):
      cs = arcpy.SearchCursor(feature_pop)
      linha = cs.next()
      conta = 0
      while linha:
         conta += 1
         linha = cs.next()
```

```
      return conta
def main_ArcGIS(quartel, custo_hospital, s_custo, populacao, field_pop, limites,
saida, rel, momento):
      w = "C:\\areatrab"
      os.mkdir(w)
      cst_distance(s_custo, quartel, w + "\\rstDist_quartel.tif")
      SecX10ToMinutes(w + "\\rstDist_quartel.tif", w + "\\min_quartel.tif", s_custo)
      SecX10ToMinutes(custo_hospital, w + "\\min_saude.tif", s_custo)
      soma_raster(w + "\\min_saude.tif", w + "\\min_quartel.tif", w + "\\soma.tif",
      s_custo)
      tempEnvironment0 = arcpy.env.extent
      arcpy.env.extent = s_custo
      if momento == "1":
         arcpy.gp.Reclassify_sa(cria_layer(w + "\\soma.tif", "soma", True), "Value", "0 15
         1; 15 30 2; 30 45 3; 45 60 4; 60 90 5; 90 120 6; 120 500 7", saida)
      elif momento == "2":
         arcpy.gp.Reclassify_sa(cria_layer(w + "\\soma.tif", "soma", True), "Value", "60
         90 1; 90 120 2; 120 150 3; 150 180 4; 180 240 5; 240 1500 6", saida)
      arcpy.env.extent = tempEnvironment0
      cria_campo(saida, momento)
      id_classes = RasterToPolygon(saida, "classe", limites, w)
      arcpy.CreateRandomPoints_management(w, "pnt_pop.shp", cria_layer(populacao,
      "grelha", False), "", field_pop, "5 Meters", "POINT", "0")
      excel = xlwt.Workbook()
      add_folha = excel.add_sheet('perfil_acesso')
      colunas = ['Nr_Individuos', 'Perc_Individuos']
      for i in range(len(colunas)):
         add_folha.write(0, i + 1, colunas[i])
      lyr_isocronas = cria_layer(w + "\\d_isocronas.shp", "isocronas", False)
      lyr_pop = cria_layer(w + "\\pnt_pop.shp", "individuos", False)
      tot_individuos = conta_pop(lyr_pop)
      cursor = arcpy.SearchCursor(lyr_isocronas)
      linha = cursor.next()
      i = 1
      ls = []
      arcpy.env.workspace = w
      while linha:
         classe = int(linha.getValue("GRIDCODE"))
         shp_classe = "classe_" + str(classe) + ".shp"
         expressao = "GRIDCODE = " + str(classe)
         arcpy.Select_analysis(lyr_isocronas, shp_classe, expressao)
         shp_clip = "clipPop_" + str(classe) + ".shp"
         arcpy.Intersect_analysis([shp_classe, lyr_pop], shp_clip, "", "", "")
         cs = arcpy.SearchCursor(shp_clip)
         ln = cs.next()
         conta = 0
         while ln:
            conta += 1
            ln = cs.next()
         for chave in id_classes.keys():
            if classe == chave:
               add_folha.write(i, 0, id_classes[chave])
               add_folha.write(i, 1, conta)
               add_folha.write(i, 2, (conta / float(tot_individuos)) * 100.0)
         i += 1
         ls.append(conta)
         linha = cursor.next()
      soma = 0
      for e in ls:
         soma = e + soma
      add_folha.write(i, 0, "Total")
      add_folha.write(i, 1, soma)
      add_folha.write(i, 2, (soma / float(tot_individuos)) * 100.0)
      excel.save(rel)
```

```
      try:
         shutil.rmtree(w)
      except:
         print "A pasta nao foi apagada"
main_ArcGIS(bombeiros, cst_saude, sup_cst, pop_grid, nr_pop, lmt,
cst_dist_reclss, relatorio, teste)
```

Programa 45 - Algoritmo (em Python, v. 2.7.4) que se propõe a gerar um cartograma de isócronas que representa o tempo (em cada célula) que alguém, caso solicitasse ajuda a partir desse local, teria de esperar até ser assistido e transportado para a unidade de saúde mais próxima, e que quantifica o número de indivíduos que estão a uma determinada distância dos equipamentos de prestação de socorro e da unidade saúde mais próxima – GRASS GIS 7.0.0

```
import os, shutil, xlwt, random
from osgeo import ogr
def v_in_ogr(shapefile, grass_vector):
      from grass.pygrass.modules import Module
      adiciona = Module("v.in.ogr", input=shapefile, output=grass_vector, flags='o',
      overwrite=True, run_=False)
      adiciona()
def r_in_gdal(raster, grass_raster):
      from grass.pygrass.modules import Module
      adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
      overwrite=True, run_=False)
      adiciona()
def r_cost(custo, orig, saida):
      from grass.pygrass.modules import Module
      acmu_cst = Module("r.cost", input=custo, output=saida, start_points=orig,
      run_=False)
      acmu_cst()
def algebra(expressao, saida):
      from grass.pygrass.modules import Module
      calc = Module("r.mapcalc", saida + " = " + expressao, overwrite=True, run_=False)
      calc()
def reclss(raster, timing, saida):
      os.mkdir("C:\\temporario")
      txt = open("C:\\temporario\\regras.txt", "w")
      dic = {}
      if timing == 1:
         txt.write("0 thru 15  = 1\n")
         txt.write("15 thru 30  = 2\n")
         txt.write("30 thru 45  = 3\n")
         txt.write("45 thru 60  = 4\n")
         txt.write("60 thru 90  = 5\n")
         txt.write("90 thru 120  = 6\n")
         txt.write("120 thru 500  = 7")
         for i in range(7):
            if i + 1 == 1:
               dic.update({1:"entre0e15min"})
            elif i + 1 == 2:
               dic.update({2:"entre15e30min"})
            elif i + 1 == 3:
               dic.update({3:"entre30e45min"})
            elif i + 1 == 4:
               dic.update({4:"entre45e60min"})
            elif i+1 == 5:
               dic.update({5:"entre60e90"})
            elif i+1 == 6:
               dic.update({6:"entre90e120"})
            elif i+1 == 7:
```

```
                dic.update({7:"maior120"})
        elif timing == 2:
            txt.write("60 thru 90  = 1\n")
            txt.write("90 thru 120  = 2\n")
            txt.write("120 thru 150  = 3\n")
            txt.write("150 thru 180  = 4\n")
            txt.write("180 thru 240  = 5\n")
            txt.write("240 thru 2000  = 6")
            for i in range(6):
                if i + 1 == 1:
                    dic.update({1:"entre60e90min"})
                elif i + 1 == 2:
                    dic.update({2:"entre90e120min"})
                elif i + 1 == 3:
                    dic.update({3:"entre120e150min"})
                elif i + 1 == 4:
                    dic.update({4:"entre150e180min"})
                elif i+1 == 5:
                    dic.update({5:"entre180e240"})
                elif i+1 == 6:
                    dic.update({6:"maior240"})
        txt.close()
        regras = "C:\\temporario\\regras.txt"
        from grass.pygrass.modules import Module
        reclassify = Module("r.reclass", input=raster, output=saida, rules=regras,
        overwrite=True, run_=False)
        reclassify()
        shutil.rmtree("C:\\temporario")
        return dic
def CreateRandomPoint(minX, maxX, minY, maxY):
        x = minX + (random.random() * (maxX - minX))
        y = minY + (random.random() * (maxY - minY))
        return [x, y]
def create_random_points(shape_pop, field):
        driver = ogr.GetDriverByName("ESRI Shapefile")
        shp = driver.Open(shape_pop)
        layer = shp.GetLayer()
        individuos = []
        for feature in layer:
            nr_pop = int(feature.GetField(field))
            if nr_pop > 0:
                geom = feature.GetGeometryRef()
                x = []
                y = []
                for polygon in geom:
                    lista = str.split(str(polygon), ",")
                    c = 0
                    if lista[0][0] == "L":
                        for string in lista:
                            if c == 0:
                                temp = str.split(string, "(")
                                par = temp[1]
                            elif c >= 1 and c <len(lista)-1:
                                par = string
                            elif c == len(lista) -1:
                                temp = str.split(string, ")")
                                par = temp[0]
                            c += 1
                            coord = str.split(par, " ")
                            x.append(float(coord[0]))
                            y.append(float(coord[1]))
                    elif lista[0][0] == "P":
                        figura = str(polygon)
                        nova = str.split(figura[10:-2], ",")
```

```
                for par in nova:
                    coord = str.split(par, " ")
                    x.append(float(coord[0]))
                    y.append(float(coord[1]))
            min_x = float(min(x))
            max_x = float(max(x))
            min_y = float(min(y))
            max_y = float(max(y))
            for i in range(nr_pop):
                individuo = CreateRandomPoint(min_x, max_x, min_y, max_y)
                individuos.append(individuo)
    return individuos
def create_features(lista, saida):
    driver = ogr.GetDriverByName("ESRI Shapefile")
    output = driver.CreateDataSource(saida)
    out_layer = output.CreateLayer(saida, geom_type=ogr.wkbPoint)
    featureDefn = out_layer.GetLayerDefn()
    for coord in lista:
        feature = ogr.Feature(featureDefn)
        ponto = ogr.Geometry(ogr.wkbPoint)
        ponto.AddPoint(coord[0], coord[1])
        feature.SetGeometry(ponto)
        out_layer.CreateFeature(feature)
    output.Destroy()
def RasterToPolygon(raster, vector):
    from grass.pygrass.modules import Module
    raster2poly = Module("r.to.vect", input=raster, output=vector, type='area',
    column='value', overwrite=True, run_=False)
    raster2poly()
def extract(vector, name_output, classe):
    from grass.pygrass.modules import Module
    onde = "value = " + str(classe)
    extrai = Module("v.extract", input=vector, type="area", where=onde,
    output=name_output, overwrite=True, run_=False)
    extrai()
def intersection(pontos, poligonos, saida):
    from grass.pygrass.modules import Module
    intersect = Module("v.select", ainput=pontos, atype='point', binput=poligonos,
    btype='area', output=saida, operator='intersects', overwrite=True, run_=False)
    intersect()
def conta_numero_pnt(entrada):
    from grass.pygrass.vector import VectorTopo
    layer = VectorTopo(entrada)
    layer.is_open()
    layer.exist()
    layer.mapset
    layer.open(mode='r')
    try:
        linha = layer.next()
    except:
        return
    conta = 0
    try:
        while linha:
            conta += 1
            linha = layer.next()
    except:
        layer.close()
    return conta
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
```

```
def __main__(quartel, custo_hospital, sup_custo, populacao, field_pop, limites,
saida, rel, momento):
     v_in_ogr(quartel, "quartel")
     r_in_gdal(sup_custo, "sup_custo")
     r_cost("sup_custo", "quartel", "cstDist_emerg")
     r_in_gdal(custo_hospital, "cstDist_saude")
     algebra("((cstDist_saude / 10.0) / 60.0) + (cstDist_emerg / 60.0)",
     "sum_isocronas")
     id_classes = reclss("sum_isocronas", 1, "rcls_isocronas")
     individuos = create_random_points(populacao, field_pop)
     os.mkdir("C:\\temporario")
     create_features(individuos, "C:\\temporario\\pop_aleatorio.shp")
     v_in_ogr("C:\\temporario\\pop_aleatorio.shp", "individuos")
     try:
        shutil.rmtree("C:\\temporario")
     except:
        print "Temporario nao foi apagado"
     RasterToPolygon("rcls_isocronas", "v_isocronas")
     lst_extract = []
     for categoria in id_classes.keys():
        extract("v_isocronas", "cat_" + str(categoria), categoria)
        lst_extract.append("cat_" + str(categoria))
     c = 1
     lst_intersect = []
     for i in range(len(lst_extract)):
        intersection("individuos", lst_extract[i], "intersect_" + str(c))
        lst_intersect.append("intersect_" + str(c))
        c += 1
     excel = xlwt.Workbook()
     add_folha = excel.add_sheet('perfil_acesso')
     colunas = ['Nr_Individuos', 'Perc_Individuos']
     for i in range(len(colunas)):
        add_folha.write(0, i + 1, colunas[i])
     total = conta_numero_pnt("individuos")
     c = 0
     for chave in id_classes.keys():
        individuos = conta_numero_pnt(lst_intersect[c])
        add_folha.write(chave, 0, id_classes[chave])
        add_folha.write(chave, 1, indivíduos)
        add_folha.write(chave, 2, (individuos / float(total)) * 100.0)
        c += 1
     excel.save(rel)
     r_out_gdal("rcls_isocronas", saida)
```

Programa 46 – Bloco de código (em Python, v. 2.7.4) que incrementa o procedimento que visa a aplicação do método de transformação/fusão IHS através da utilização das ferramentas *i.rgb.ihs* e *i.ihs.rgb* do GRASS GIS 7.0.0.

```
def r_in_gdal(raster, grass_raster):
     from grass.pygrass.modules import Module
     adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
     overwrite=True, run_=False
     adiciona()
def rgb2ihs(i_red, i_green, i_blue):
     from grass.pygrass.modules import Module
     transf = Module("i.rgb.his", red=i_red, green=i_green, blue=i_blue, hue="i_hue",
     intensity="i_intensity", saturation="i_saturation", overwrite=True, run_=False)
     transf()
def ihs2rgb(i_hue, i_saturacao, i_intensidade):
     from grass.pygrass.modules import Module
```

```
    trans = Module("i.his.rgb", hue=i_hue, intensity=i_intensidade,
    saturation=i_saturacao, red="t_red", green="t_green", blue="t_blue", overwrite=True,
    run_=False)
    trans()
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
def ihs(vermelho, verde, azul, pan, out_red, out_green, out_blue):
    r_in_gdal(vermelho, "vermelho")
    r_in_gdal(verde, "verde")
    r_in_gdal(azul, "azul")
    r_in_gdal(pan, "pancromatica")
    rgb2ihs("vermelho", "verde", "azul")
    ihs2rgb("i_hue", "i_saturation", "pancromatica")
    r_out_gdal("t_red", out_red)
    r_out_gdal("t_green", out_green)
    r_out_gdal("t_blue", out_blue)
```

Programa 47 - Bloco de código (em Python, v. 2.7.4) que incrementa o procedimento que visa a aplicação do método de transformação/fusão IHS através da utilização da ferramenta *i.pansharpen* do GRASS GIS 7.0.0.

```
def r_in_gdal(raster, grass_raster):
    from grass.pygrass.modules import Module
    adiciona = Module("r.in.gdal", input=raster, output=grass_raster, flags='o',
    overwrite=True, run_=False)
    adiciona()
def r_out_gdal(grass_raster, saida):
    from grass.pygrass.modules import Module
    output = Module("r.out.gdal", input=grass_raster, output=saida, format='GTiff',
    overwrite=True, run_=False)
    output()
def ihs_pansharpen(vermelho, verde, azul, pancroma, saida, workspace):
    r_in_gdal(vermelho, "vermelho")
    r_in_gdal(verde, "verde")
    r_in_gdal(azul, "azul")
    r_in_gdal(pancroma, "pancromatica")
    from grass.pygrass.modules import Module
    output = Module("i.pansharpen", red="vermelho", green="verde", blue="azul",
    pan="pancromatica", output=saida, method="ihs")
    r_out_gdal(saida + "_red", workspace + "\\pansharped_red.tif")
    r_out_gdal(saida + "_green", workspace + "\\pansharped_green.tif")
    r_out_gdal(saida + "_blue", workspace + "\\pansharped_blue.tif")
```

ANEXO B – INQUÉRITO POR QUESTIONÁRIO PARA RECOLHA DE DADOS SOBRE ADOPÇÃO DE *SOFTWARE* LIVRE E/OU DE CÓDIGO ABERTO NA ADMINISTRAÇÃO PÚBLICA

Mestrado em Tecnologias de Informação Geográfica – Universidade de Coimbra

Breve Nota de Enquadramento:

Este Inquérito constitui uma ferramenta de trabalho que, sendo elaborado no âmbito do desenvolvimento de uma Dissertação de Mestrado em Tecnologias de Informação (FLUC/FCTUC), tem única e exclusivamente propósitos e finalidades científicas e académicas. Esta Dissertação apresenta-se como um exercício que surge da necessidade de se diagnosticar dificuldades e avaliar factores como o grau de conhecimento e a sensibilização para a possibilidade/necessidade/obrigatoriedade de utilizar Software Livre/de Código Aberto (SLCA) nos organismos da Administração Pública Portuguesa (APP), designadamente, Software SIG/TIG. Instrumentos como, por exemplo, a Lei nº 36/2011 (Normas Abertas) e o OE2013 vieram introduzir uma mudança de paradigma na utilização de serviços e plataformas informáticas na APP que urge, de facto, avaliar em termos práticos, no âmbito dos critérios e parâmetros a ter em conta no processo de tomada de decisão: "migrar ou não migrar para Software Livre / de Código Aberto". Este é o *leit motiv* do pedido de colaboração que apresentamos na bateria de questões que estruturam este Inquérito. Bem-hajam!

Parte I – Perfil do Inquirido:

1.1. Sexo: _____

1.2. Idade: _____

1.3. Habilitações literárias (grau académico e área de formação): _____

_____

No serviço/organismo/instituição/empresa refira:

1.4. Qual o cargo/categoria profissional que ocupa?

_____

1.5. Há quanto tempo? _____

1.6. Qual o papel/função/competências que desempenha na condição referida em 1.4?

_____

1.7. Há quanto tempo? _____

1.8. A sua experiência profissional na área das TIC e/ou das TIG:

i) É inferior a um ano ☐ ii) Tem entre 1 e 3 anos ☐ iii) É superior a 3 anos ☐

1.9. Utiliza Software Livre/de Código Aberto, em concreto, software SIG?

Sim ☐ Não ☐

1.10. Em caso de resposta afirmativa à questão anterior indique o nº de anos. _____

Parte II – Perfil da Organização:

2.1. Designação do serviço/organismo/instituição:

_____

2.2. Número estimado de trabalhadores que trabalham com soluções SLCA? _____

2.3. Número estimado de trabalhadores que trabalham com soluções SLCA?

i) Software, em geral ☐ ii) Software SIG/TIG ☐

2.4. A sua organização já implementou ou está a implementar algum tipo de solução baseada em SLCA, designadamente, na área dos SIG/TIG?

Sim ☐ Não ☐

2.5. Em caso de resposta negativa à questão anterior, os elementos com poder de decisão estão ao corrente da importância do SLCA e das normas que regulamentam a sua utilização na APP?

Sim ☐ Não ☐

2.6. Em sua opinião existe sensibilização dos seus colaboradores para aprender a trabalhar com SLCA?

Sim ☐ Não ☐

2.7. Estão a ponderar introduzi-lo em algum processo emergente?

Sim ☐ Não ☐

2.8. No seu serviço/organismo/instituição existem recursos humanos com capacidade técnica e conhecimento para fazer formação interna no âmbito da utilização de SLCA, ou seria necessário recorrer a formação externa (onerosa)?

_____

_____

_____

_____

2.9. No caso de no seu serviço/organismo/instituição se utilizar já algum tipo SLCA qual é o grau de satisfação com a solução informática encontrada?

i) Satisfação total iii) Satisfação parcial, mas o balanço é negativo

ii) Satisfação parcial, mas o balanço é positivo iv) Insatisfação total

2.10. No caso de no seu serviço/organismo/instituição já se utilizar algum tipo de SLCA qual é o espectro de aplicação deste tipo de solução? É usada para um conjunto de problemas menores ou garante suporte a, pelo menos, um serviço crítico, tendo um âmbito alargado de utilização?

_____

_____

_____

_____

2.11. Considera o processo de migração efectuado pelo seu organismo como um exemplo de sucesso?

Sim ☐ Não ☐

Justificação:

_____

_____

_____

_____

2.12. Considera que, na sua organização, o pessoal ao serviço mostra-se pouco interessado em migrar/utilizar SLCA?

Sim ☐ Não ☐

2.13. Em caso de resposta afirmativa à questão anterior, indique as 3 razões invocadas com maior frequência como justificação para a resistência à mudança?

_____

_____

_____

_____

Parte III – Visão do Inquirido sobre o processo de migração entre soluções e constituição de um manual de apoio à decisão que valide e oriente o primeiro:

A tomada de decisão relativa ao processo de adopção de SLCA na AP depende de um balanço que cada serviço/organismo/instituição deve fazer internamente, o qual terá de tomar em consideração a relação (quociente) entre o Total Cost of Ownership (TCO) e o Return on Investment (ROI).

3.1. Em sua opinião, quais deverão ser os parâmetros a incluir em cada uma destas rúbricas?

_____

_____

_____

_____

3.2. Em sua opinião, quais são as principais dificuldades/condicionantes que se apresentam como obstáculo à realização de um balanço desta natureza?

_____

_____

_____

_____

3.3. Considera que a construção de um **manual/guia de orientação** seria útil para a tomada de decisão sobre o processo de migração para SLCA na APP, na medida em que permitisse inventariar e avaliar os parâmetros a ter em conta nessa equação?

Sim ☐ Não ☐

No âmbito do presente trabalho de dissertação, a elaboração deste ***guia/manual*** *será precedida por um conjunto de exercícios de modelação espacial que têm como objectivo comparar diferentes soluções (proprietárias e livres), de modo a serem avaliados o nível de confiança e viabilidade de cada uma das diferentes alternativas. Tratando-se de um* ***guia/manual*** *que obriga a reflectir sobre o dilema da migração de software em organismos da APP, julga-se por avisado promover um inventário de situações-problemas reais. Assim sendo:*

3.4. Na sua instituição de acolhimento profissional, com que tipo de problemas espaciais lidam diariamente e quais são os software SIG/TIG utilizados para a resolução desses mesmos problemas (pode alargar a sua resposta a outros tipos solucionados com SLCA)?

_____

_____

3.5. Que outro tipo de recomendações lhe parecem merecer nota digna de destaque no âmbito da elaboração de um manual/guia de orientação para a tomada de decisão sobre o processo de migração para SLCA na APP?

_____

_____

_____

_____

Bem-haja pela atenção dispensada!

ANEXO C – INVENTÁRIO DE SOFTWARE LIVRE E/OU DE CÓDIGO ABERTO PARA APLICAÇÃO NA ÁREA DAS TIG

Nome	Versão actual	Endereço URL	Sistema Operativo	Linguagem programação
SIG *Desktop*				
DIVA-GIS	7.5	http://www.diva-gis.org/	Mac OS, Windows	Java
Flowmap	7.4.2	http://flowmap.geo.uu.nl/	Windows	?
Fmaps	0.0.2	http://fmaps.sourceforge.net/#Introduction	Linux	C
Generic mapping tools	5.1.2	http://gmt.soest.hawaii.edu/projects/gmt	Mac OS, Windows, Linux	C/C++
ILWIS	3.8	http://52north.org/	Windows	?
GRASS GIS	7.0	http://grass.osgeo.org/	Mac OS, Windows, Linux	C, Python
gvSIG	2.2	http://www.gvsig.com:9090/pt	Mac OS, Windows, Linux	Java
HidroSIG	4.0	http://www.medellin.unal.edu.co/~hidrosig/index.php?option=com_content&view=article&id=1&Itemid=28&lang=en	Windows, Linux	Java
ILWIS	3.8	http://52north.org/	Windows	?
Kalypso	15.1.1	http://kalypso.bjoernsen.de/	Windows, Linux	?
KOSMO	2.0.1	http://www.opensig.es/index.php	Mac OS, Windows, Linux	Java
MapWindow GIS	4.8.6	http://www.mapwindow.org/	Windows	C++, C#
OpenJUMP	1.8.0	http://openjump.org/	Mac OS, Windows, Linux	Java
OrbisGIS	5.0.1	http://orbisgis.org/	Mac OS, Windows, Linux	Java
QGIS	2.10.1	http://qgis.org/en/site/	Mac OS, Windows, Linux	C++
SAGA GIS	2.2.2	http://www.saga-gis.org/en/index.html	Windows, Linux	C++
SAMT	4.0	http://www.zalf.de/en/forschung/institute/lsa/forschung/methodik/samt/Pages/default.aspx	?	C++
SavGIS	?	http://www.savgis.org/	Windows	?
SPRING	5.3	http://www.dpi.inpe.br/spring/english/index.html	Windows, Linux	?

Thuban	1	http://thuban.intevation.org/	Mac OS, Windows, Linux	Python
uDIG	1.5.0	http://udig.refractions.net/	Mac OS, Windows, Linux	Java
Bibliotecas de algoritmos geoespaciais e *toolbox* incorporáveis em software SIG *desktop*				
Agent Analyst	1	http://resources.arcgis.com/en/help/agent-analyst/	ArcGIS	Jython
CGAL	4.7	http://www.cgal.org/	MacOS, Windows, Linux	C++
FDO	4.0.0	http://fdo.osgeo.org/	?	?
FWTools	2.4.7	http://fwtools.maptools.org/	Windows, Linux	Java, Python
GDAL/OGR	2.0.1	http://www.gdal.org/	Windows, Linux	C/C++
Generic Geometric Library		http://trac.osgeo.org/ggl/		C++
GeoOxygene	1.7	http://oxygene-project.sourceforge.net/	MacOS, Windows, Linux	Java
Geospatial Modelling Environment	0.7.4	http://www.spatialecology.com/gme/	ArcGIS	?
GeoKettle	2.5	http://www.spatialytics.org/projects/geokettle/		
GEOS	3.5.0	https://trac.osgeo.org/geos/	MacOS, Windows, Linux	C++
GeoTools	13.5	http://geotools.org/		Java
libLAS	1.8.0	http://www.liblas.org/	MacOS, Windows, Linux	C/C++
MGET	0.8a60	http://mgel.env.duke.edu/mget	ArcGIS	C, MATLAB, R, Python
OpenMap	5.0.3	http://openmap-java.org/	Mac OS, Windows, Linux	Java
Patch Analyst	5	http://www.cnfer.on.ca/SEP/patchanalyst/Patch5_1_Install.htm	ArcGIS	
Proj.4		https://trac.osgeo.org/proj/		C++
Puzzle GIS		https://www.openhub.net/p/puzzle-gis		
Sextante	?	?	?	?
SharpMAP	1.1	http://sharpmap.codeplex.com/	Indep	C#
Spatial Data Modeller		http://www.ige.unicamp.br/sdm/	ArcGIS	Python
TauDEM	5	http://hydrology.usu.edu/taudem/taudem5/index.html	ArcGIS/ GDAL	
TerraLib		http://www.terralib.org/		C++, Java, Visual Basic
Sistema de Gestão de Base de Dados				
PostgreSQL PostGIS	9.4.5 2.2.0	http://www.postgresql.org/	Mac OS, Windows, Linux	Java, C
Web SIG				

Deegree	3.3.17	http://www.deegree.org/	MacOS, Windows, Linux	Java
Geomajas		http://www.geomajas.org/	MacOS, Windows, Linux	Java
GeoMoose	2.8.1	http://www.geomoose.org/	MacOS, Windows, Linux	Java
GeoServer	2.8.1	http://geoserver.org/	MacOS, Windows, Linux	Java
istSOS	2.2.1	http://istsos.org/	MacOS, Windows, Linux	Python
Mapbender		http://mapbender3.org/	MacOS, Windows, Linux	PHP, Java
MapFish	2.2	http://www.mapfish.org/	MacOS, Windows, Linux	Java
MapGuide	3.0	http://mapguide.osgeo.org/	Windows, Linux	PHP, Java
MapServer	7.0.0	http://www.mapserver.org/	MacOS, Windows, Linux	PHP, Python, Perl, C#, Java
OpenLayers	3.10.1	http://openlayers.org/	MacOS, Windows, Linux	Java
PyWPS	3.2.2	http://pywps.wald.intevation.org/	MacOS, Windows, Linux	Python
Team Engine		https://github.com/opengeospatial/teamengine	MacOS, Windows, Linux	Java
ZOO-Project	1.5.0	http://www.zoo-project.org/	MacOS, Windows, Linux	PHP, Python, Perl, C#, Java
Detecção Remota e Fotogrametria				
E-foto	2015.09.385	http://www.efoto.eng.uerj.br/	Windows, Linux	?
InterImage	1.43	http://www.lvc.ele.puc-rio.br/projects/interimage/pt-br/	Windows, Linux	?
Opticks	4.12	http://opticks.org/confluence/display/opticks/Welcome+To+Opticks	Windows, Linux	?
Orfeo ToolBox Monteverdi2	5.0.0 0.8.1	https://www.orfeo-toolbox.org/	Windows, Linux	C++
OSSIM	1.8.20	http://trac.osgeo.org/ossim/	Mac OS, Windows, Linux	C++