Eduardo Jorge do Val Martins

CHOPIN-CCO
Command Center of Operations for Search and Rescue Missions with Human and Robotic Teams

June 2014

· U C ·

Universidade de Coimbra

UNIVERSITY OF COIMBRA

FACULTY OF SCIENCES AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTERS ENGINEERING

CHOPIN-CCO
Command Center of Operations for Search and Rescue Missions with Human and Robotic Teams

Eduardo Jorge do Val Martins

A dissertation presented for the degree of Master of Science

in Electrical and Computers Engineering

Coimbra, 2014

· U C ·

CHOPIN-CCO Command Center of Operations for Search and Rescue Missions with Human and Robotic Teams

Supervisor:

Prof. Dr. Paulo Menezes

Co-Supervisor:

Prof. Dr. Rui Paulo Rocha

Jury:

Prof. Dr. Jorge Lobo

Prof. Dr. Paulo Coimbra

Prof. Dr. Paulo Menezes

Report written for the dissertation subject, included in the Electrical and Computers Engineering Course, submitted in partial fulfillment for the degree of Master of Science in Electrical and Computers Engineering.

Coimbra, 2014

“I see a strong parallel between the evolution of robot intelligence and the biological intelligence that preceded it. The largest nervous systems doubled in size about every fifteen million years since the Cambrian explosion 550 million years ago. Robot controllers double in complexity (processing power) every year or two. They are now barely at the lower range of vertebrate complexity, but should catch up with us within a half century.”

Hans Moravec

Acknowledgements

I take this opportunity to express my special appreciation and thanks to my supervisor, Prof. Dr. Paulo Menezes and co-supervisor, Prof. Dr. Rui Rocha, for their expert knowledge, guidance, monitoring and constant encouragement throughout the course of this thesis.

I would like to thank to all the firefighters, particulary the Bombeiros Sapadores de Coimbra, and members of the Autoridade Nacional de Protecção Civil, for the valuable information provided by them in their respective fields, specially to Comandante António Lousada from Bombeiros Voluntários da Mealhada, without whose time and effort would have been more difficult to understand the firefighters concept of operations. I am grateful for their cooperation during the period of this thesis work.

I also thank to all my colleagues of the CHOPIN project for their valuable ideas and helpful suggestions that have indicated new directions in the development of the project, and also to my laboratory colleagues for the stimulating discussions and insightful comments to overcome technical implementation issues.

Lastly, I specially thank my parents for their constant encouragement without which this thesis would not be possible.

Abstract

The use of ubiquitous computing to help firefighters in the Urban Search and Rescue (USAR) missions, has been studied in the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) project, with the aim of exploring cooperation between human and robotic teams in the context of these missions, decentralizing some decisions so that they can be made closer to the place to which they relate, based on existing information.

In this context, this dissertation presents an architecture for the USAR mission system and develops an interface for the Command Center of Operations (”Centro de Comando de Operações”) (CCO), integrating the context-aware software platform with the required tools for planning and monitoring USAR missions, leaving the detection and interpretation of data to other computing entities on the ground, thus enhancing situational awareness and establishing a common ground to support decision making of the tactical actors.

The interface is developed by addressing interaction design techniques to fulfill *situation-awareness* and human-robot interaction requirements, in order to support collaboration between human and robotic teams, thus proposing a message taxonomy to provide information interchange among agents, displaying and registering the collected data on the ground by individual agents, such as maps, locations and status of several features as victims and fire outbreaks, as well as environmental conditions in the Theatre of Operations (”Teatro de Operações”) (TO) in a relational database.

The overview of the operations on the ground is presented using the services provided by the Google Maps Application Programming Interface (API), allowing an overall picture of the TO, existing resources and the state of the mission.

The sharing of information between the CCO and agents on the ground is made via a bidirectional and asynchronous WebSocket, and is based on the Robot Operating System (ROS) framework and a pre-defined set of messages to share information cooperatively by firefighters and robotic agents to fulfill the mission, using the format JavaScript Object Notation (JSON). The compression of transmitted information proved critical to communications with limited bandwidth and high traffic volumes, usually found in USAR scenarios.

Keywords: Search and rescue, command center of operations, ROS, interaction design, JSON, context-aware, situation-awareness, human-robot interaction.

Resumo

A utilização da computação ubíqua para ajudar os bombeiros nas operações de busca e salvamento em cenários urbanos, tem vindo a ser estudada no projecto de I&D Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN), para explorar a cooperação entre equipas humanas e robóticas no âmbito destas missões, descentralizando algumas decisões para que eles possam ser tomadas mais próximas do local em que se referem, com base nas informações existentes.

Neste contexto, esta dissertação apresenta uma arquitetura para o sistema de apoio de missões de B&S e desenvolve um interface para o CCO, integrando na plataforma informática, sensível ao contexto, as ferramentas necessárias para o planeamento e monitorização das missões de B&S, deixando a detecção e interpretação de dados para outros sistemas computacionais no terreno, aumentando assim a percepção do contexto e estabelecendo uma base comum para apoiar as tomadas de decisão dos atores estratégicos.

O interface é desenvolvido abordando técnicas de design da interação e orientado por requisitos para a percepção da situação e requisitos de interação homem-robô, para apoiar a colaboração entre equipas humanas e robóticas, propondo uma taxionomia de mensagens para permitir a partilha de informações entre os agentes, exibindo os dados recolhidos no terreno pelos agentes individuais, tais como mapas, localização e estado de pontos de interesse como vítimas e focos de incêndio, bem como condições ambientais no Teatro de Operações (TO) e registando os dados numa base de dados relacional.

A visão geral das operações no terreno é apresentada recorrendo aos serviços disponibilizados pela API do Google Maps, permitindo uma imagem global do TO, dos recursos existentes e do estado da missão.

A partilha de informação entre o CCO e os agentes no terreno é efectuada de forma bidireccional e assíncrona através de um WebSocket, tendo por base a framework ROS e um conjunto de mensagens pré-definidas, para partilhar informações de forma cooperativa pelos bombeiros e agentes robóticos no cumprimento da missão, utilizando o formato JavaScript Object Notation.

A compressão da informação transmitida revelou-se fundamental para comunicações com largura de banda limitada e elevado volume de tráfego, características dos cenários urbanos de B&S.

Palavras-Chave: Busca e salvamento, centro de comando de operações, ROS, design da interação, JSON, sensibilidade ao contexto, percepção da situação, interacção homem-robô.

Contents

List of Figures

List of Tables

Code Excerpt

Acronyms

ACD	Activity-Centered Design
ANPC	National Authority for Civil Protection (”Autoridade Nacional de Protecção Civil”)
API	Application Programming Interface
BLEVE	Boiling Liquid Expanding Vapor Explosion
BS	Base Station
BSD	Berkeley Software Distribution
C2	Command and Control
CC	Command Center
CCO	Command Center of Operations (”Centro de Comando de Operações”)
CDOS	District Command (”Comando Distrital de Operações de Socorro”)
CECOP	Operations Section (”CÉlula de Combate/OPerações”)
CELOG	Logistics Section (”CÉlula de LOGística”)
CEPLAN	Planning Section (”CÉlula de PLANeamento”)
CHOPIN	Cooperation between Human and rObotic teams in catastroPhic INcidents
CODIS	(”COmandante DIStrital de Operações de Socorro”)
COPE	Common Operational Picture Exploitation
COS	Incident Commander (”Comandante das Operações de Socorro”)
CSCW	Computer-Supported Cooperative Work
CTA	Cognitive Task Analysis
DMS	Degree, Minutes, Seconds
ECMA	European Computer Manufacturers Association
ENB	Firefighters National School (”Escola Nacional de Bombeiros”)

ERD	Entity-Relationship Diagram
FPS	First-Person Shooter
GIS	Geographic Information System
GPL	General Public License
GPS	Global Positioning System
GDTA	Goal Directed Task Analysis
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HMI	Human Machine Interface
HRI	Human-Robot Interaction
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISR	Institute of Systems and Robotics ("Instituto de Sistemas e Robótica")
JSON	JavaScript Object Notation
LGPL	Lesser General Public License
LOA	Level of Automation
LODD	Line Of Duty Death
MANET	Mobile ad hoc Network
MRL	Mobile Robotics Laboratory
MRSLAM	multi-robot simultaneous localization and mapping
NOP	Permanent Operational Rule ("Norma Operacional Permanente")
OS	Operating System
PCO	Command Operations Station ("Posto de Comando Operacional")
POSIT	Situational report ("POnto de SITuação")
RFC	Request for Comments

ROS	Robot Operating System
SA	Situation Awareness
SADO	Operational Decision Support System ("Sistema de Apoio à Decisão Operacional")
SC	Sector Commanders
SCBA	Self-Contained Breathing Apparatus
SGML	Standard Generalized Markup Language
SGO	Operations Management System ("Sistema de Gestão de Operações ")
SIRESP	Portuguese Integrated Security and Emergency Network System ("Sistema Integrado das Redes de Emergência e Segurança de Portugal")
SITAC	Tactical Situation (SItuação TÁCtica)
SOG	Standard Operating Guideline
SOP	Standard Operating Procedure
SQL	Structured Query Language
TCP	Transmission Control Protocol
TETRA	Terrestrial Trunked Radio
TO	Theatre of Operations ("Teatro de Operações")
UCD	User-Centered Design
USAR	Urban Search and Rescue
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition

Chapter 1

Introduction

This document describes the work done during the dissertation project "CCO - Command Center of Operations for Search and Rescue Missions with Human and Robotic Teams", integrated into the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) project, developed in the Mobile Robotics Laboratory (MRL) of the Institute of Systems and Robotics ("Instituto de Sistemas e Robótica") (ISR), University of Coimbra. The main goal for this master degree dissertation is to develop the software for the Command Center of Operations ("Centro de Comando de Operações") (CCO), to plan, control and monitor an Urban Search and Rescue (USAR) mission presenting a global vision of the state of the mission on the ground and individual state of the agents and resources available on the Theatre of Operations ("Teatro de Operações") (TO).

This work aims to explore cooperative architectures to decentralize some of the decisions, on the basis of local information, demanding they be made as close as possible to the place where it relates, speeding up communications between agents and with the CCO. Despite this decentralized approach, the main center of the decision, at a higher level of abstraction, and the place where all available information about the mission is collected and monitored, remains in the CCO.

This chapter presents not only the context and motivation, but also the main goals of this dissertation and how it is organized.

1.1 Context and Motivation

Firefighters face great risks in USAR missions due to the uncertainty of the scenarios where they operate. In a dynamic stressful environment, the tasks required to a team can change depending on the team's situational context, forcing the team members to adapt [SSB04].

Decision making is largely impacted by the external environment, and the decision maker is affected by several factors, e.g. *time stress* which results in fatigue and loss of vigilance, forcing the decision maker to use less complicated reasoning strategies in order to save time.

Despite the evolution in information and communication technologies, and the studies in pervasive computing, firefighters on ground level still use archaic methods to register incident occurrences, depending on the expertise and resilience of the decision makers. Some steps have been taken and Portugal adopted the *Portuguese Integrated Security and Emergency Network System ("Sistema Integrado das Redes de Emergência e Segurança de Portugal")* (SIRESP) system to integrate the communications of all civil defense agencies, and the *District Command ("Comando Distrital de Operações de Socorro")* (CDOS) of *National Authority for Civil Protection ("Autoridade Nacional de Protecção Civil")* (ANPC) is equipped with an *Operational Decision Support System ("Sistema de Apoio à Decisão Operacional")* (SADO) which does not give direct support to firefighters and commanders on the field.

Firefighter Fatality Investigation & Prevention Program of NIOSH[1] identifies *inadequate risk assessment*, *lack of accountability or SCBA air management (BA Control)*, *inadequate communication* and *lack of situational awareness* as some of the most common factors associated with Line Of Duty Death (LODD). Other study by the IAFF[2] (USA) quantifies *lack of situational awareness* as responsible for 37.3% firefighters' injuries [SOP09], therefore, the use of technology to enhance situational awareness and to automate and control time-depending tasks, by alerting the decision-makers can save firefighters lives.

The use of robot teams in cooperation with human teams for search and rescue operations allows decisions to be made closer to the place to which they relate, based on information provided by various sensors of the robots and the human team experience. Robotic teams can assist human teams in USAR missions, browsing in dangerous areas and with low visibility looking for survivors, providing important information to define the context in which the teams operates, allowing more rapid, efficient and safer operations for the human teams and allowing the CCO to anticipate hazardous events for agents on the ground.

[1]National Institute for Occupational Safety & Health
[2]International Association of Fire Fighters

However, the whole operation will always be planned, monitored and controlled from the Command Center (CC), so this work aims to develop a computer interface capable of providing an overview of events in the theatre of operations, monitoring the status, location and actions of all agents operating in the terrain, as well as the logistical resources mobilized within the mission, providing this central node with all information available to support the commander decisions.

This interface aims to automate some tasks that are nowadays carried out manually by the commander, including recording and monitoring information coming from every agent in the TO, and generating operational warnings when critical situations are detected. In this way, the person in charge of the mission can avoid being distracted with collecting and organizing such information, being focused in analyzing it and making decisions.

1.2 Objectives

The objectives of the present work are to design a system to integrate firefighters' operations and robot teams on the TO, defining the requirements of the Human Machine Interface (HMI) to plan, control and monitor search and rescue actions, providing the decision maker with the perception of the running events on the ground, thus enhancing Situation Awareness (SA) and leading to a higher level of cognitive control over the situation.

The design must also support SA for multiple and distributed operators, providing collaboration among agents, by integrating different technologic platforms and definition of a message taxonomy for data-interchange.

The cooperative exchange of information in order to decentralize some decisions is achieved by using the latest technologies, such as fixed large touch screens, hand held devices, internet access, and robots with several sensors, communicating over a Mobile ad hoc Network (MANET) providing best-effort communication to the search and rescue teams.

Due communication constrains in unpredictable and dynamic environments, tag mission information for later retrieval represents a relevant feature to define context in time and space, therefore, main events of USAR operations are registered in a database for prior context definition during ongoing actions and subsequent analysis of the course of the mission.

1.3 Approach followed in this dissertation

Interaction design is a formal discipline to support the design of products to support people in their everyday and working lives [RSP11]. The objective of interaction design is to ensure that a product is usable, useful and the taken actions make sense to the users.

Interaction design follows one of four approaches: *user-centered design*, *activity-centered design*, *systems design* and *genius design*. While *user-centered design* is focused on translating the user needs and goals, which is helpful to improve acceptance of the end users; *activity-centered design* approach is focused on tasks and activities that have to be accomplished, allowing high level of task automation, but leading sometimes to poorly designed systems, hard to learn and operate; *systems-design* is focused on the components of the system, using the whole context to provide a holistic approach to system design, deemphasizing the users' skills; and finally *genius design* relies on the skill and wisdom of the designer to make design decisions, involving the final user at the end of the process, to ensure the system works as the designer has predicted.

In an collaborative environment, as USAR missions, technology plays a central role in establishing a knowledge common ground based on ubiquitous computing, enhancing SA for the CCO and teams on the ground. [EBJ03] proposed a user-centered design methodology to determine SA requirements conducted by a form of cognitive task analysis, designated as Goal Directed Task Analysis (GDTA), based on the goals and decisions of the systems' operators, providing a flexible Level of Automation (LOA), by enhancing the information that should appear in the displays to support collaboration among them and decision making.

To understand the concept of operations and model first responders operations on the ground, several meetings were promoted with the domain experts, firefighters' corporations and members of the National Authority for Civil Protection ("Autoridade Nacional de Protecção Civil") (ANPC) structure: *Bombeiros Sapadores de Coimbra*, *Bombeiros Voluntários da Mealhada* and *Bombeiros Sapadores de Faro*; with the district command of ANPC: *Paulo Palrilha, 2.º ("COmandante DIStrital de Operações de Socorro") (CODIS) de Coimbra* and *José Bismark, CODIS de Aveiro*; and documentation review of the manuals of the Firefighter National School (Firefighters National School ("Escola Nacional de Bombeiros") (ENB)), ANPC standard operating procedures (NOPs) and Portuguese legislation.

The gathered material allows to identify the mission goals, the tasks performed by first responders and acting patterns in unpredictable environments, leading to the elicitation of user requirements and the design of user interfaces, supported by interaction design principles and *Goal Directed Task Analysis* (GDTA) .

The integration of firefighters operations and the use of ubiquitous computing in the scope of the CHOPIN project is presented in Figure 1.1, where the context of this dissertation is delimited by the red box.

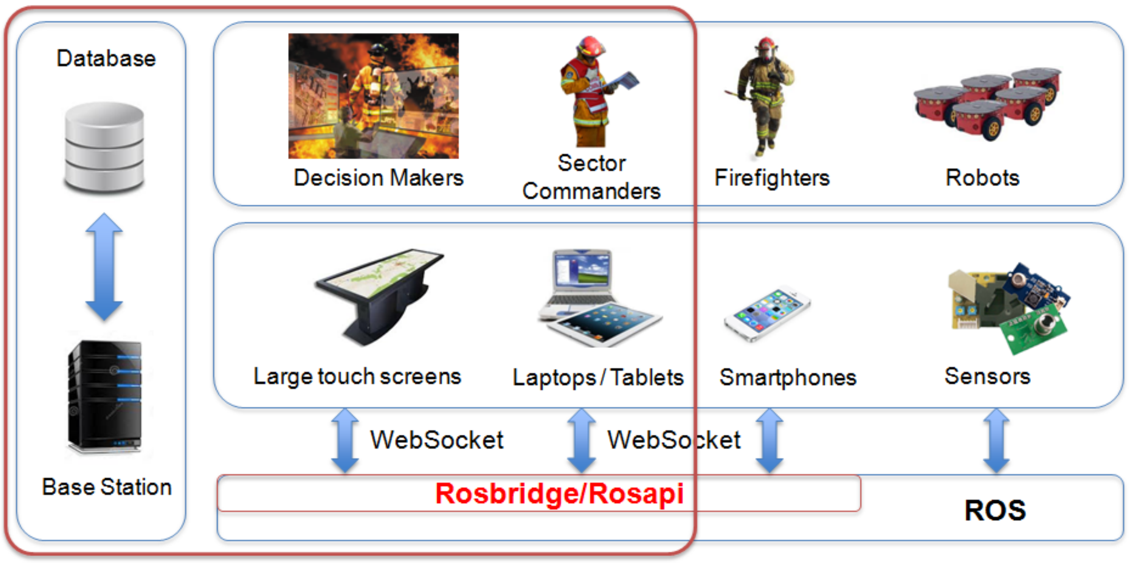


FIGURE 1.1: Scope of the project.

The integration of technologies introduced by the CHOPIN project with CCO were tested in controlled laboratory environment, namely local maps transmission, robots' pose, environmental variables readings, victims and fire-outbreaks location and firefighters status.

1.4 Outline of the dissertation

This document is organized in six chapters and four appendices. The first chapter introduces the context and motivation, the objectives and the approach followed to attain the objectives, using *interaction design* techniques.

In chapter 2, firefighters actions methodology in USAR missions and the used documentation are analyzed to specify the first requirements of the system (top-level requirements), and give an overview of the product main functions and the definition of *use cases*, thus presenting the main classes of the system. The operating environment is defined and characterized.

Chapter 3 analyzes the interface requirements for the CCO system, addressing interaction design methodologies, context-aware and the Human-Robot Interaction constraints to determine the overall Human Machine Interface requirements with top-level decision makers and human/robotic agents on the ground, connected through a proposed message taxonomy to support the information flow between the system participants. It is also presented the hardware and software interfaces required by the proposed system.

The designed solutions of the HMI interface are presented in chapter 4, along with a description of the integration of HTML5 and JavaScript with the *Qt* framework for the graphical tactical situation; the data interchange between CCO and Robot Operating System (ROS), for publishing and subscribing topics; techniques tested for occupancy-grid transmission to the CCO; the firefighters and robots features accepted by the system; and the gesture recognition feature provided by the application.

Chapter 5, introduces the database requirements analysis and design, using a clustering technique, to register the chain of main events in the incident, in order to support SA tagging.

Chapter 6 presents the results of this dissertation, describing the steps taken by the tactical actors, in a simulated incident, with several images of the proposed interface, and the relation between pages.

Finally, Chapter 7 presents the main conclusions of this dissertation and provides directions for future work within the scope of the CHOPIN project.

Chapter 2

Requirements Analysis

2.1 Application Context

Operational model and procedures of civil defense agencies and fire departments of different countries are similar. Usually, the incident response mission is triggered by an emergency call to the control centre which mobilizes first responders to the incident zone. The first responders' commander starts to gather as much information as possible about the incident (reconnaissance phase) and defines an appropriate response tactics. On field level, the command post is settled and the incident commander must get an understanding of the amount of additional resources and the level of command required [NOP12] [GHC09].

2.1.1 Operation Mode of Portuguese Firefighters

The smallest operational unit of the portuguese firefighters is the ***Team***, which is composed by five or six firefighters, where the ***team commander*** is the firefighter with highest rank [Des08].

The structure of the command station is evolutive, and changes as the incident takes other proportions and evolves to a different phase, thus changing the structure of command and the layout of resources on the ground [NOP12]. The phasing of the incident depends on the complexity and number of resources involved. In phase *I*, for non complex operations, up to six teams are involved, independently of their typology. In phase *II*, with resources

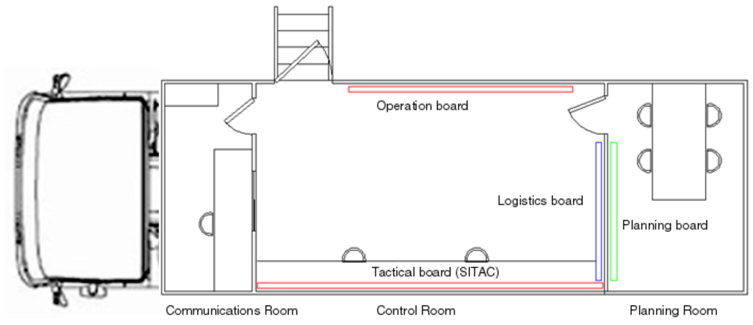


Figure 2.1: Layout of the CC vehicle.

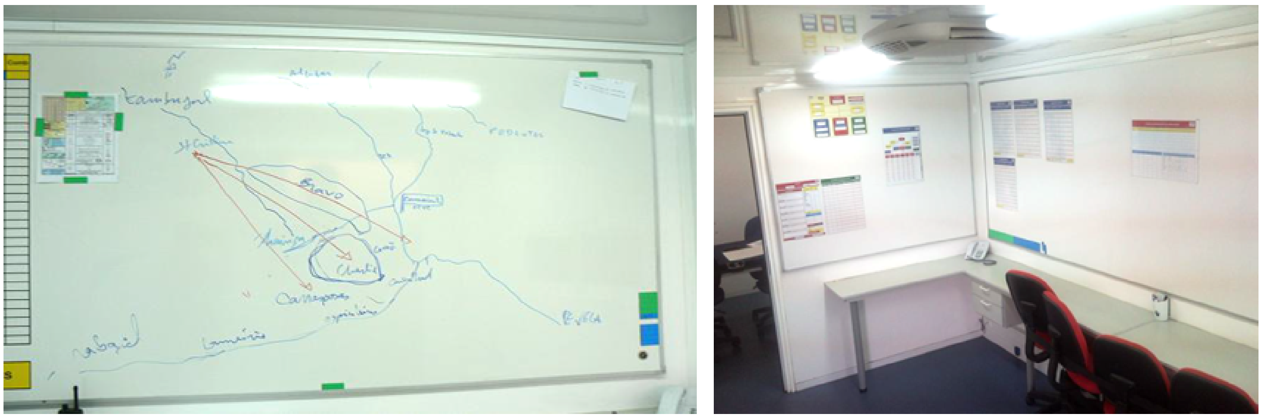
equivalent to three groups, with six teams each, the Operations Section ("CÉlula de Combate/OPerações") (CECOP) is activated. For phases *III* and *IV*, the Operations Management System ("Sistema de Gestão de Operações ") (SGO) is divided into three sections:

- Planning Section ("CÉlula de PLANeamento") (CEPLAN)
- Logistics Section ("CÉlula de LOGística") (CELOG)
- Operations Section ("CÉlula de Combate/OPerações").

This structure is used in complex and large catastrophic scenarios and, in this last phase, operations are conducted by a member of the National Authority for Civil Protection ("Autoridade Nacional de Protecção Civil") (ANPC) structure. The Command Center (CC) uses nowadays a vehicle with magnetic white boards to manually record all incidents on the ground [Figure 2.1], coordinate the operations of the Urban Search and Rescue (USAR) teams and manage the resources on the Theatre of Operations ("Teatro de Operações") (TO).

As the level of command changes, the new Incident Commander ("Comandante das Operações de Socorro") (COS) should identify clearly and unambiguously all set of resources, structures, type of ongoing and planned actions on the TO, therefore ANPC adopted a common operational symbology for the management of USAR operations [Form D.9]. These tools allow representing graphically and dynamically all main events and key information concerning the ongoing occurrence, and understanding the transmission of operational information at different levels of command, control and execution (strategic, tactical and maneuver levels)[NOP09b].

All the planned and ongoing actions, as well as teams and resources (planned and available), are represented in a white board designated by Tactical Situation (SItuação TÁCtica)



(a) SITAC on magnetic board in CCO (b) Forms on magnetic boards in CCO

Figure 2.2: SITAC main board and forms to register incidents on the ground in the PCO.

(SITAC) [Form D.5], which is updated in the course of operations providing mission global awareness.

Other predefined forms are used to record the structure of command, damages, victims, communications plan, mobilized resources and weather reports distributed through the respective boards in the CC, as shown in Figure 2.2.

All communications are centralized in the CC, based on analog radio communications and on the Portuguese Integrated Security and Emergency Network System ("Sistema Integrado das Redes de Emergência e Segurança de Portugal") (SIRESP), which is based on the European Terrestrial Trunked Radio (TETRA) communication system. Therefore, the update of the boards relies on information that is requested to agents on the ground and on their reply.

Commanders on the field rely on information to take the proper actions to control the incident and save lives of civilians and firefighters. The information that reaches the commander normally exists in large quantity and quality, but may not arrive in time, due to limitations of communications equipment, existence of shadow zones, and the accumulated stress of the team commanders. This time delay, as well as the amount of information the CCO have to process, may prevent the right decisions in the right time, thus putting at risk the mission or delaying predefined tactical actions to be taken.

2.1.2 Decision making

Command and Control (C2) in firefighters operations is based on a hierarchical military model with predefined acting rules. Nonetheless, if the decision maker is an expert, he/she

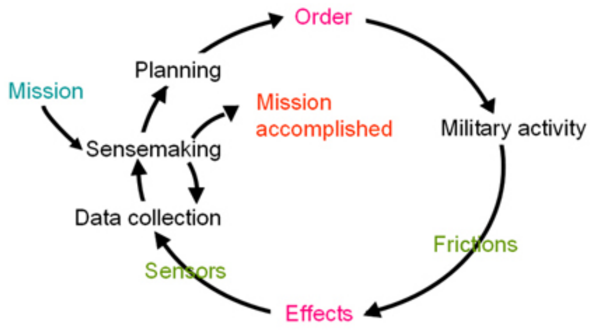


Figure 2.3: The dynamic OODA loop (DOODA loop). functions are given in black, products in red, and inputs in green

interacts with the situation at a *skill-level* of cognitive control, where the decisions are generated *subconsciously*.

At a *rule-based* level of cognitive control, the decision maker is familiar with the situation, forming his decision on prior experiences or learned from another person's know-how, where the control of behavior at this level is goal-oriented, knowing that other alternatives of action are available.

On a novel or unfamiliar situation, the decision maker acts on a *knowledge-based* (also called *model-based*) level of cognitive control, involving the transformation of declarative knowledge into procedural knowledge [Ras93].

The command and control model in a USAR mission is similar to the one used in military actions, Boyd's OODA loop (*observe* - *orient* - *decide* - *act*). Brehmer [Bre05] propose a work flow model to describe how members of the command teams perform their tasks, merging the OODA loop and cybernetic approaches to Command and Control [Figure 2.3].

The data is collected from visual contact, information received from firefighters and survivors and from the system. The mission is planned based on gathered information, pre-defined USAR actions and the commander experience, transmitting the orders to accomplish the mission and waits feedback effects to reformulate the plan of action until the mission is completed.

In dynamic and unpredictable environments such as USAR missions, all three levels of control can interact with the situation depending on the decision maker expertise and his/her training level.

2.1.3 Risk Assessment

One of the main problems that firefighters face in USAR missions, is the rapid fire phenomenon, which has three basic types, with accepted scientific terminology: *Flashover*, *Back draught* and *Fire Gas Ignition*, which conducts to high rising temperatures in a small period of time, injuring and killing nearby firefighters. Then, becomes of utmost importance the recognition of the warning signs of these impending events [SOG06], sharing the information with the surrounding agents so they can try to leave the hazard area.

Some of these warnings comprise changes in smoke conditions, like color alterations (particularly darkening), high velocity of smoke through an exit, sudden lowering layer of smoke and pulsing (raising & lowering cycle in the layer); heat alterations forcing agents to crouch low; detection of hazard conditions with inflammable gases. These changes in the environment can be detected by fusing several sensors readings of different systems, like robots and firefighter's wearable devices, augmenting situation awareness.

The three main tactics used in preventing or counter rapid fire development are *tactical ventilation*, *fire confinement* and *3D water-fog tactics*, whilst the use of these tactics are decided in the CCO, incident commander should make his/her decision based on the better knowledge of the context where the teams are working in.

2.1.4 Situation Awareness

Generally, Situation Awareness (SA) is a human's understanding of what is happening around the human operator at the current point in time and in the near future. Endsley *et al.* [EBJ03] divides the formal definition of SA into three separate levels:

- Level 1 - *perception* of the elements in the environment;
- Level 2 - *comprehension* of the current situation;
- Level 3 - *projection* of the future status.

During the decision making process, is of utmost importance that decision makers obtain high levels of situation awareness. Figure 2.4 demonstrates the relation between the decision making process and the SA levels proposed by Endsley *et al.* [EBJ03].

USAR missions occur in highly dynamic and unpredictable environments where the situation is always changing forcing operator's SA to change constantly or become inaccurate. Human

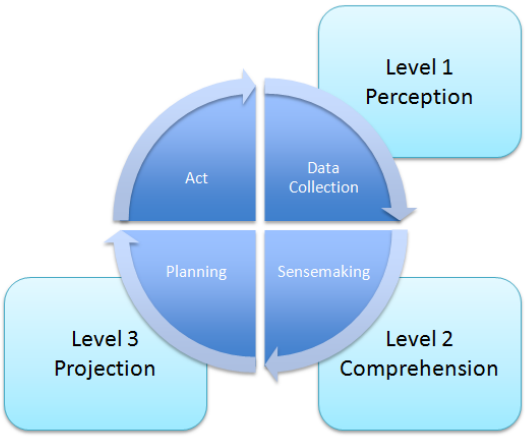


FIGURE 2.4: DOODA loop functions and Situation-awareness levels

operators have limited attention capacity to attend to multiple items simultaneously, and Miller [Mil56] states that only seven (plus or minus 2) chunks of unrelated information can be held and manipulated in humans' working memory, which consist in two major constraints for SA.

In these dynamic environments, where information changes constantly and is hard to obtain, other factors can deteriorate significantly SA, like *attentional-tunnelling*, *data overload*, *workload*, *anxiety*, *fatigue*, *misplaced salience*, *errant mental models* and *out-of-the-loop syndrome*.

The system design must take these factors into account to avoid them, create and maintain high levels of SA and the fact that decision makers form mental models of how the system works to form higher levels of SA (comprehension and projection) providing default information even when data is incomplete or missing.

For a stronger situational awareness, the messages passed among agents and the CCO should be clear to influence the effectiveness of information.

2.2 User Scenarios

The analysis of the organization of the management operations system of firefighters in search and rescue missions, and integrating the robots in operations to assist decision makers and firefighters in their actions on the ground increasing situational awareness, leads to the definition of the *high-level requirements* of the CCO and the conceptual domain model [Figure 2.5].

Use Case 1 - Tactical mode and risks assessment

Upon arrival, the ***incident commander*** characterize the ***incident***, access the risks in order o take proper actions, fill the adequate ***identification form*** and write a ***Situational report ("POnto de SITuação") (POSIT)*** . From the analysis of the collected data , incident's commander define the objectives of the mission and the tactical mode for each sector/team and communicate it to the Support Officer, the Operations Commander and/or the Sectors commanders. The Operations commander assigns the define tasks to the teams and the Support Officer updates the incident status to the ANPC control center [Figure A.2].

Use Case 2 - Theatre of Operations organization

The level of command is settle by the ANPC or according with rules in the NOPs, and the incident commander establishes the ***incident areas***, composed by the ***inner area*** and the ***outer area***, with a traffic point to control the entrances in the TO and inform the public. The commander defines the incident sectors around the incident core and deploys the ***resources*** in the defined areas. These operations are registered in the ***SITAC board*** with appropriate graphical symbology.

As the incident phase changes, the system changes the structure of the incident alerting the commander defines establishes the PCO, composed by three cells (Operational Cell, Planning Cell and a Logistic Cell with a Concentration and Reserve Area) [Figure A.1].

Use Case 3 - Search and Rescue

The incident commander assigns the tactical goals for search and rescue victims, from the collected data, deploying the robots in the SITAC board and sending a signal to start the mission. If a victim is found the robots communicate the occurrence to the CCO, with the register in the ***Victims Board***, and to the nearby agents which have assigned mission tasks classified as non-critical and if they are suitable equipped regarding the environmental classification, to carry out the rescue operation, the system inform the incident commander about the situation and suggests options, so the proper actions can be taken, as assign a team to proceed with the rescue operation and mobilizing an available ambulance or requesting a new one to the Control Center [Figure A.7].

Use Case 4 - Fire attack

The Operations commander (COS) evaluates the situation and defines the tactical goals of the mission briefing the firefighters and deploying the robots. The commander defined tactics may include a fire exposure protection, to protect the nearby structures, an exit protection so the firefighters and victims can move away from the incident area, an offensive or defensive fire attack tactic or eventual tactical ventilation, which is registered in the SITAC board. The commander or the user designated by him/her operates a large touch screen with combined views of the actions on the ground.

Use Case 5 - Manage personnel

The incident commander assign the tactical goals of the mission and brief the crew in a clear manner to maintain SA, monitors the crew deployment and the progress of the agents on the ground. The operations commander updates the tactical goals for the sectors commanders, so the defined tactics can be implemented by the operational teams. The command support officer updates the status of the personnel to the ANPC control center.

Use Case 6 - Manage resources

The incident commander mark available sources of water in the surroundings and request new sources of water to provide a continuous flow of water supply. All the vehicles and other equipments are placed on the map, requesting more resources if the existing ones are insufficient. This data is updated by the operations commander in response of updates demand supply of the sectors commanders, and forwarded to the ANPC control center by the command support officer.

The domain classes of the system are depicted in *italic* and presented in the *domain model* in Figure 2.5

2.3 User Classes

The use cases in Appendix A demonstrate the distinct roles of robots, firefighters and commanders within the system. These diagrams also mention the various boards which are updated during the operations. The strategic and tactical commanders are outside the hot zone with access to these boards controlling the progress of ground operations. According to their function and location, systems' user classes are identified.

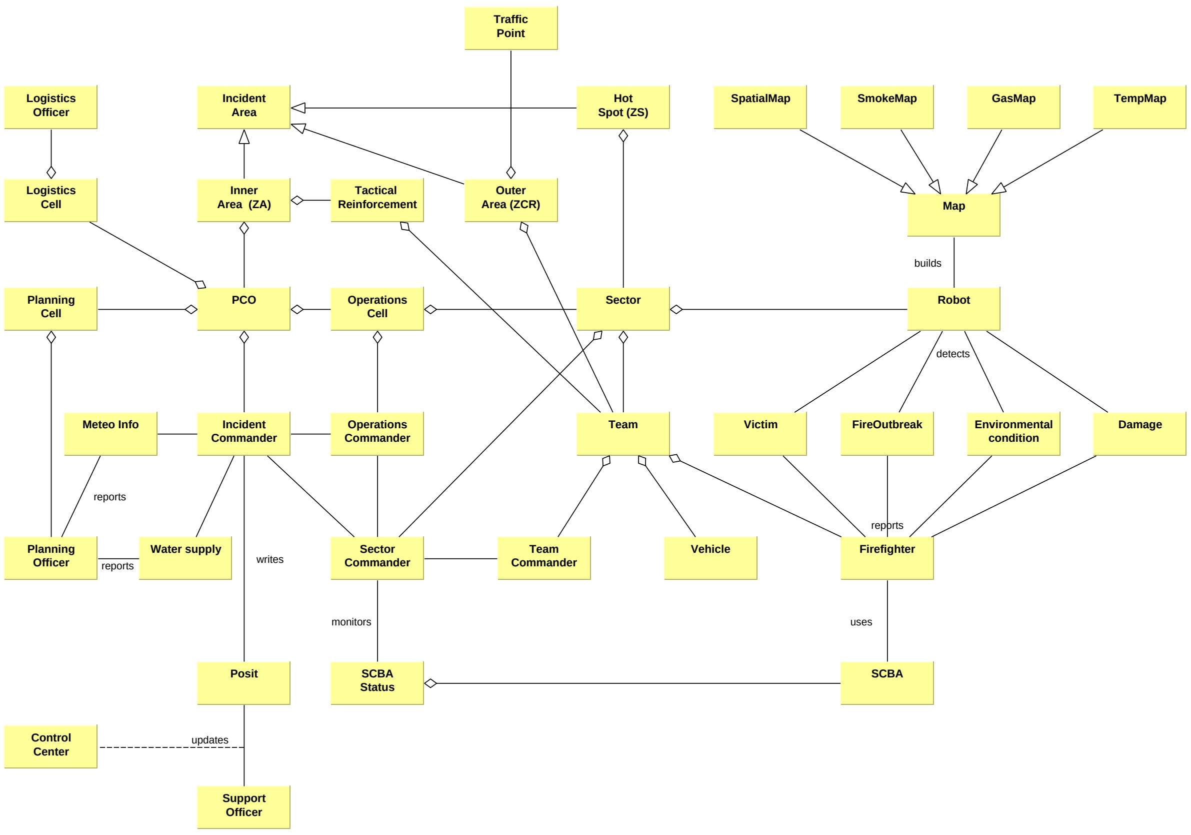


FIGURE 2.5: Conceptual domain model of incident operations

- The ***incident commander*** and the ***supporting officers*** are located in the same physical space with access to fixed large touch screens, sharing the same information.
- The ***sector commanders*** are located within the limits of the accident area, monitoring the course of actions within its sector, using tablets connected to the network to interact with the system and the agents on the ground.
- The ***breath apparatus entrance control officer*** coordinates the Self-Contained Breathing Apparatus (SCBA) activities, when adverse conditions in the building require a tight entrance control to limit the time of exposure to the incident. This can be achieved by registering SCBA activities and the firefighters information in the respective board with a tablet connected to network. SCBAs with level control are too expensive and the activities must be time controlled, echoing an alarm, when the defined limit is reached, in the firefighters' board and in the mobile device attached to the firefighter.
- ***Firefighters*** operate inside the sinister area, to search and rescue potential victims, determine hazardous conditions and inform the COS about their actions and course of

the mission. The firefighters use a mobile device with sensors, e.g. smart phone, connected to the system by a Mobile ad hoc Network (MANET) to update the firefighter status and send/receive some small messages and signals (e.g. if the firefighter stay still for more than thirty seconds an alarm will echo to warn the command center and the other firefighters nearby.

- ***Robots*** provide crucial information, building temperature, gas and 2D maps and identifying potential victims and hazardous conditions for the agents on the ground. This information is registered in the robots' board, the SITAC board, victims' board and the operations board. Robots are connected to the Base Station (BS) through the MANET and communicate with the CCO using a predefined set of messages.
- ANPC control center receives request for new resources and status reports updates through the support commander officers and is external to the system.

2.4 Operating Environment

USAR missions operates in extreme conditions, under severe interferences caused by the surrounding environment, with communications with low signal strength, the communications between the nodes (CCO, firefighters and robots) relies on a MANET, to ensure data transmission, taking advantage of the proximity between agents on the ground, bridging through neighbors to reach the message destiny.

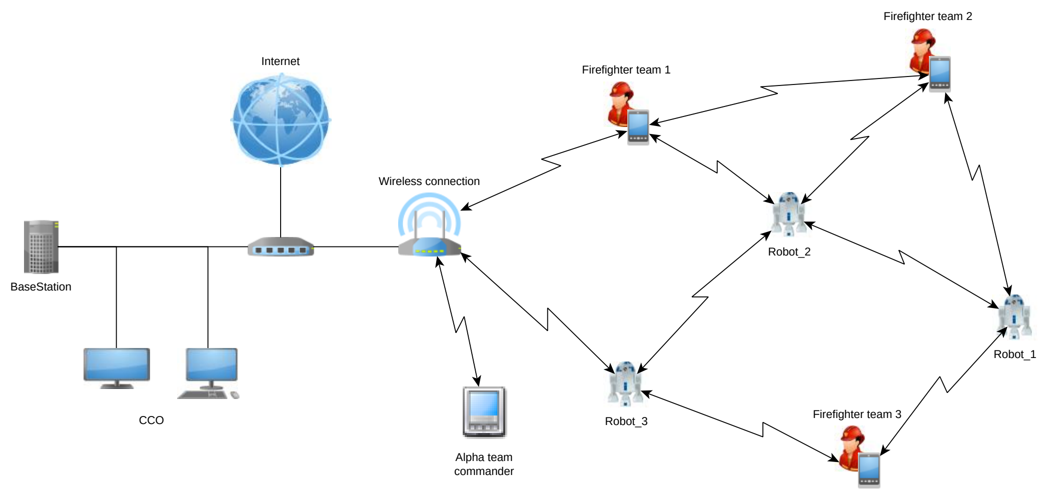


FIGURE 2.6: System architecture

Robots use Robot Operating System (ROS) to communicate, publishing and subscribing topics to share information about the surrounding environment and interacting with the firefighters on the ground. ROS framework provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality and message-passing between processes.

Interconnection between these different systems relies on Rosbridge which is an applications layer network protocol specification and allows the non-ROS participants to publish and subscribe topic messages and invoke services running on the base station using JavaScript Object Notation (JSON)-formatted messages over Transmission Control Protocol (TCP) web sockets.

Web socket is a protocol providing full-duplex communications channels over a single TCP connection. It provides a way for the server to send unsolicited data to the client, and allowing for messages to be passed back and forth while keeping the connection open, which is suitable for to publish/subscribe ROS topics messages.

The system relies on a BS where all services are running, the ROS framework (RosCore, RosBridge and RosAPI) and MySQL server to store all the incident main events. The CCO can be installed in the BS, or in another device, connected via Ethernet to the BS. Base Station operating system must support the ROS framework which is primarily tested on Linux and Mac OS X systems.

The information sharing between human and robotic teams and the command center is extremely important and it is required to use a common base for validation and communication between the used platforms like JSON which is an open standard format that uses human-readable text to transmit data, language-independent, used by Rosbridge as message format and it's possible to take advantage of *Qt* native parser/serializer to decode/encode topic messages.

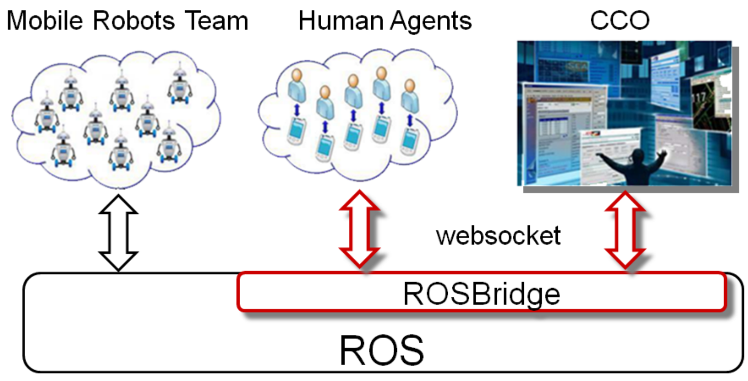


Figure 2.7: Communication structure of the different actors.

2.5 Summary

This chapter started by defining the context where the application is going to operate, analyzes the operation model of the portuguese firefighters, with the elicitation of technologies and equipments they use, introducing some constraints related to the decision-maker tasks, the risk-assessment necessary to preserve human lives and property, addressing the bases of the necessary common operational picture exploitation to prevent the deterioration of situational-awareness of the end-users. Hereafter, the analysis of the possible user scenarios in USAR operations provides guidance to the identification of the user classes of the system, that together with the characterization of the operating environment leads to the definition of the interface requirements presented in the next chapter.

Chapter 3

Interfaces Requirements

3.1 Hardware Interfaces

The interaction interface is supported by the hardware interface where several views of the Human Machine Interface (HMI) are distributed over fixed touch screens for the tactical operators, mobile devices (e.g. iPADs) for the Sector Commanders (SC), smart phones (e.g. iPhones) for the firefighters and multiple sensors (lasers, sonars, dust, alcohol and pyroelectric sensors) for the robots on the Theatre of Operations (”Teatro de Operações”) (TO). All these devices can be used to share information and implement collaborative context awareness.

The use of computers, tablets and smart phones increases the ability to transmit other data types, complementing the standard audio channel used in search and rescue scenarios. Robots provide important information for strategic decision and enable a better context definition. The visual information for representing tasks status; planning and ongoing actions are transmitted more accurately than verbal communication [Ash07].

Adding ”Touch” technologies functionality provides a more intuitive operation mode. Moreover, the development of online mapping websites has become popular, facilitating access to geospatial maps that were only available through a Geographic Information System (GIS), allowing a detailed view of the surrounding TO. Databases with Global Positioning System (GPS) coordinates, combined with mapping web sites, can be a helpful tool to indicate several crucial points (e.g. Hydrants) and the location of available resources within the TO.

Devices indicated here only serve as a proof of concept, and should be replaced by more rugged and appropriate devices in real Urban Search and Rescue (USAR) missions.

3.2 Software Interfaces

The Command Center of Operations ("Centro de Comando de Operações") (CCO) is developed using the *Qt* application which is a cross-platform framework that uses standard C++ and a special code generator (called the Meta Object Compiler or moc) together with several macros to enrich the language. *Qt* runs on major desktop platforms and on some of the mobile platforms, like Linux, Windows 8, iOS and Android. Other important features includes: *signals* & *slots* programming; HTML5 and *JavaScript* support, necessary to make use of the Google Maps Application Programming Interface (API); Structured Query Language (SQL) database access; eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) parsing; thread management and network support, expanding options to mainstream the different technologies needed to develop the project.

Communications between the agents on the ground and the CCO are based on *tufao* web socket, an asynchronous web framework for C++ that takes advantages of *Qt's* object communication system (*signals* & *slots*) and is free software published in terms of the Lesser General Public License (LGPL). [Oli12].

In Base Station (BS), the applications layer network protocol *rosbridge* provides a JSON API to Robot Operating System (ROS) functionalities for the non-ROS applications through the WebSocket interface *rosbridge_server*. While the CCO application gains access to ROS services calls with the package *rosapi*, to determine topic types and to request messages.

Integration of Google Maps with the application is provided by QtWebkit, which is an open source web browser engine. Its API allows *Qt* applications to render regions of dynamic web content providing a mechanism of *signals* & *slots* to communicate with the *JavaScript* loaded in the WebView widget.

3.3 Interaction design requirements

The interface design development is focused on finding a suitable representation that engages the end-user inputs to achieve the desired system output, making the application usable and useful [Saf07].

To meet these criteria the CCO interface development addresses to some interaction design laws that should be observed for a good *Human Machine Interface* (HMI) . Therefore, ***clickable*** objects, like operating buttons, are designed to have reasonable sizes and placed preferentially on the edges and corners of the screen to make it easier and faster to operate, since they are huge targets, with theoretically infinite height or width, because the mouse stops at the edge of the screen even with a fast movement, by referring to Fitts' law [Fit54] which states *"the time it takes to move from a starting position to a final target is determined by two factors: the distance to the target and the size of the target"*.

This law is a model of the human movement that predicts that the time required to move to a target area is a function of the distance to the target and the size of the target, and Equation 3.1 is the Shannon formulation of Fitt's law proposed by Scott MacKenzie [Mac92].

$$MT = a + b \times log_2 \left(1 + \frac{D}{W}\right) \tag{3.1}$$

Where MT is the movement time to the target (sec); D is the distance between the pointing device and the target (cm); W is the size (diameter) of the target (cm); $a = 0.230sec$ and $b = 0.166sec$ are constants, experimental determined and target dependent.

Gesture recognition is added to the application combined with large touch screens to meet the Fitts' law statement, because it is more natural and faster for a user to point to an object with his finger rather than with the *mouse* pointer. The selectable objects has an associated information window or pop-up menu next to them, with no more than ten options to allow a direct manipulation of the object and a faster decision of which action to take.

Hick-Hyman law [Hic52, Hym53] says that the time it takes the user to make a decision is determined by the number of options available, which is consistent with George Millers *Magical Number Seven* rule, where he states that the human mind is best able to remember information in chunks of seven pieces, plus or minus two, having some difficulties to keep more than that information amount in short-term memory [Mil56].

The design of an application to operate in a dynamic and unpredictable environment, capable of enhancing Situation Awareness (SA) and maintaining knowledge common ground for decision makers and operating agents presents large complexity, so according to Teslers law of *Conservation of Complexity* it is important to transfer some complexity to the system. Teslers law states that there is a point beyond which it's not possible to simplify a process any further, and some complexity can only be transferred from one place to another [Saf07]. This is achieved automating some processes, drawing alarm signs and eliminating redundant tasks to focus the decision makers' attention in really compelling operations.

The system interaction design must be driven by the *Poka-Yoke* principle, avoiding (*yokeru*) inadvertent errors (*poka*), to adjust some behaviors of the end users and prevent eventual errors. This principle was created in 1961 by the Japanese industrial engineer and quality guru Shigeo Shingo. To accomplish this principle the system must inform the end user after a performed action (*feedback*) to trace operation success and before an action is taken (*feedforward*) so the operator knows what is about to happen, limiting the options of the end-user.

3.4 Context-awareness requirements

The overall system design is context-aware, exploiting decentralized collaboration between human and robotic teams, sharing the surrounding context with the nearby agents and the CCO.

Abowd *et al.* [ADB+99] define *context* as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. The most relevant context types to characterize the situation of a particular entity, according to this author, are ***location***, ***identity***, ***time*** and ***activity*** defining the primary level of context, responding to the questions, ***where***, ***who***, ***when*** and ***what*** [Figure 3.1].

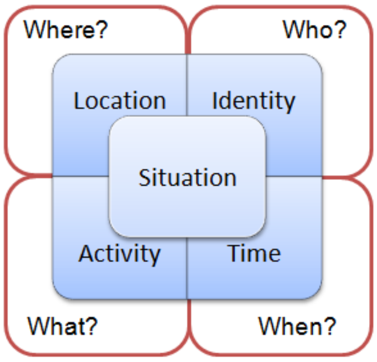


FIGURE 3.1: Context types to characterize a situation

So every feature in the system presents this primary information content to characterize the context.

The definition of Context-aware is not a consensual issue, requiring application's behavior be modified to be considered context-aware, Abowd *et al.* [ADB+99] states that "a system is ***context-aware*** if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users' task". This context-aware definition is more general, therefore applications that only displays context information about the mission's environment without modifying its behavior, are also classified as context-aware application.

Context-aware applications support features depending on the context, so Abowd *et al.* [ADB+99] propose a ***context-awareness*** taxonomy, fusing the taxonomy proposals of Schilit *et al.* [SAW94] and Pascoe [Pas98], into three categories:

1. **presentation of information and services to a user** :
   - agents and the CCO have the knowledge of discovered features locations and status, thus enhancing SA;
   - robot agents fuse data sensors readings to activate CCO and nearby agents application widgets when relevant features in the environment changes, alerting end-users in a convenient form for context alteration, reducing data overload;
   - also visible options to the end-user reflect the possible actions to be taken in accordance with the surrounding context;
2. **automatic execution of a service**:
   - using an appropriate level of automation and decentralizing some decisions from tactical actors, the system responds to particular context changes activating services, which without any reasonable doubt, the human actor should take (e.g. firefighter with a Self-Contained Breathing Apparatus (SCBA) should receive a retreat order after half time work of the used device capacity;
   - after an evacuation command, if a human agent is moving towards a blocked exit or a hazard area, agents on the ground with different SA should warn and provide an alternative exit for the firefighter);
3. **tagging of context to information for later retrieval**:
   - information context sharing provides the users with contextual augmentation with additional digital information. Environmental alterations and new discovered features are registered in the database to encompass contextual events of the surroundings and forwarded to the agents enhancing SA.

3.5 Human-Robot Interaction requirements

USAR missions are classified as ***synchronous*** (at the same time) and ***non-collocated*** (in different spaces) in the canonical Computer-Supported Cooperative Work (CSCW) Ellis *et al.* [EGR91] "Time-space taxonomy", but taxonomy does not cover the dimension of collaborators communication mode, which is an important feature in USAR missions, which can also be classified as ***collocated*** (in the same space) in our approach to collaborative work between human and robotic teams.

Yanco and Drury [YD04] present a taxonomy to classify Human-Robot Interaction (HRI) and overcome the "Time-space" and other proposed taxonomies constrains. In the scope of the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) project, this taxonomy is adapted to support the situation awareness's five components of the HMI [Figure 3.2]: human-robot, robot-human, human-human, robot-robot and humans overall mission awareness [DHYS04].

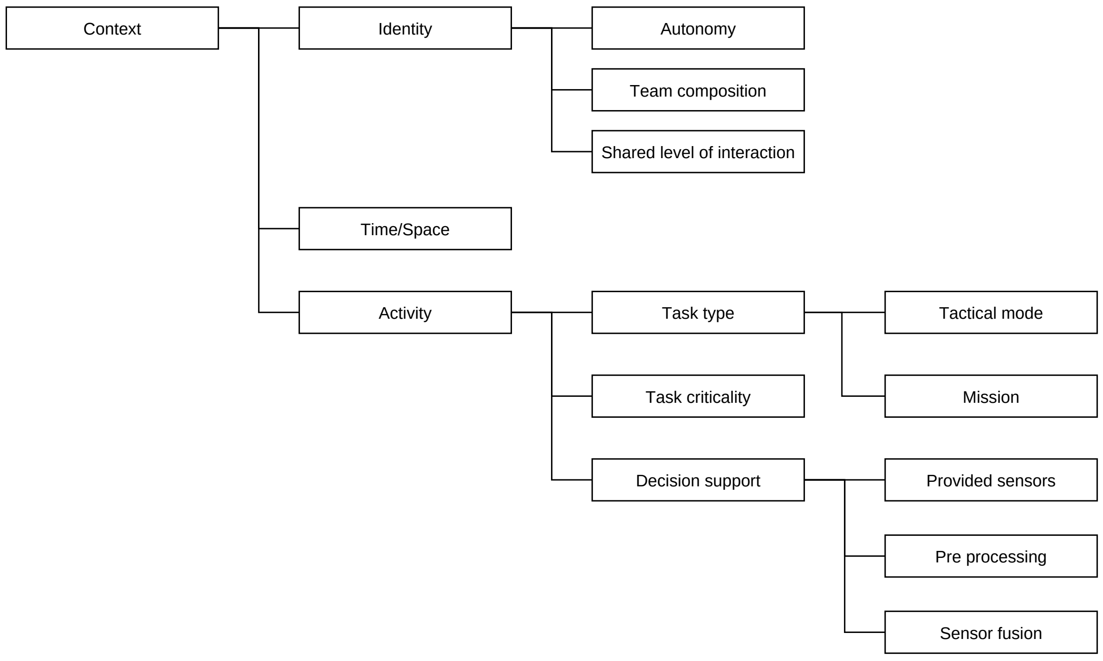


FIGURE 3.2: Taxonomy for multi-agent HRI

The relevant categories for the USAR system are:

1. ***Task Type*** [TT]: robots and firefighters on the ground perform different tasks as victim search, detection of fire outbreaks and hazard conditions and reading environmental variables, reflecting the assigned objective of the agent. These tasks have different procedures or are incompatible with the declared tactical mode

2. ***Task Criticality*** [TC]: the first aim of USAR missions is to protect human life, rescuing incident victims and assuring firefighters physical integrity, therefore some tasks have higher priority than others, by giving primacy to the protection of human life on the protection of property. Task criticality of an agent's mission can only be changed by the CCO.

3. ***Composition of Teams*** [CRT]: agents on the ground should know who their teammates are. Although they can collaborate with other team members it is important, for context characterization, that robot agents can recognize their teammates, as human agents do.

4. ***Level of Shared Interaction Among Teams*** [LSI]: in the scope of the CHOPIN project, CCO gives orders to a team of robots [Figure 3.3.B] that need robot-robot HRI awareness to coordinate themselves, although, in order to decentralize some decisions, the remain combinations are possible to occur in USAR scenarios [Figure 3.3].

5. ***Decision Support for Operators*** [DSO]: The agents on the ground should have:

   - a list of provided sensors - includes a list of sensing type to the operator for decision making.
   - a list of the type of sensor fusion, e.g. $\{\{sonar, ladar\} \rightarrow map\}$
   - the amount of pre-processing of sensors for decision support, e.g. $\{ladar \rightarrow map\}$

6. ***Time/Space*** [TS]: One of the major features to define *context* is the interveners' location and time, therefore both robots and firefighters should communicate these aspects within the incident area, during the assigned mission.

7. ***Autonomy Level*** [AL]: In this work, the level of autonomy is considered to be the capacity (battery level for robots and SCBA level for firefighters) to ensure a safe exit for the agent, considering the duration of working time, the distance to the nearest safe exit, the consumption rate and the remaining time to travel the escape path.

3.6 Human-Machine Interface requirements

The system interaction design should be directed by users' concerns to make it more attractive and easier to use (***user-centered***) but also focused on the tasks the user must accomplish (***activity-centered***).

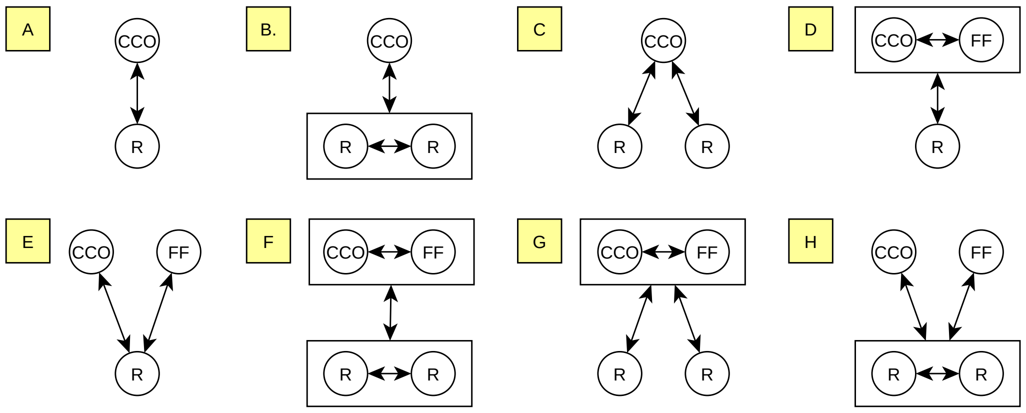


FIGURE 3.3: The possible combinations of CCO, firefighters and robots, acting as individuals or in teams. (adapted from [YD04])

Changing methodologies always find some resistance, therefore the use of the National Authority for Civil Protection (”Autoridade Nacional de Protecção Civil”) (ANPC) forms and the analysis of its usual procedures may contribute to a greater acceptance of the proposed model for search and rescue operations. Using the same symbology defined by ANPC will improve the acceptance of the system because users will feel comfortable with the vocabulary used. The graphical environment should be easy to operate, using informative icons with large dimensions representing actions, since they are better remembered than ***text commands***, allowing ***drag***&***drop*** operations, resizing and rotating objects.

The actions on the tactical situation board will have a side effect on the forms of the ANPC to create a level information automation eliminating redundant tasks and enhancing situation awareness with the use of ***dynalinks*** where information depicted in one window explicitly changes in relation to what happens in another [SR96].

Information visualization is used to amplify human cognition, enabling the users to see patterns, trends, and anomalies in the visualization [CS99] enhancing discovery, decision-making, and explanation of phenomena using touch screens with a shareable interface, providing a large interactional space that can support flexible group working and simultaneously view the interactions and have same shared point of reference as others. The use of graphical simulations - ”*provide the illusion of participating in a synthetic environment rather than external observations of such environment*” [EGJ93], to create a highly engaged user experience enhancing SA.

Endsley *et al.* [EBJ03] describe 50 principles of designing for SA, where four are directly intended to the display of information to support shared SA:

- Principle 45: Build a common picture to support team operations
- Principle 46: Avoid display overload in shared displays
- Principle 47: Provide flexibility to support SA across functions
- Principle 48: Support transmission of different comprehensions and projections across teams.

The defined HMI must have these principles into account to support shared understanding, to interact with all agents on the ground and characterize awareness. Situation awareness in this HMI has five components: ***human-robot***, ***robot-human***, ***human-human***, ***robot-robot*** and ***humans overall mission awareness*** [DHYS04].

Therefore, the HMI should:

- Provide maps of where the firefighters and robots have been, providing spatial information of the immediate surroundings, thus enhancing awareness;
- Avoid the "mental" (or manual) data fusing by human agents by providing automated multi-sensor information fusion;
- Display multiple robots on a single window, increasing efficiency and providing the Command Center (CC) operator with the most appropriate level of automation at any given time [DHYS04].

However, the use of a higher level of automation may lead to a poorer acceptance by end users. Therefore, it is important to seek a balance between human/manual and automated operation; a mixture of human and automated control is desirable [MP07].

3.6.1 Functional Requirements

The system will interact with all the actors on the ground through transactions on the screen by the end user meeting the functional requirements listed in the following tables.

Search and Rescue requirements

Table 3.1: Search and Rescue Functional Requirements

N.º	**Requirement**
SRFR 01	The system will record automatically victims discovered by the robots, as well as their location and status, with an associated warning to the end-user with a visual and acoustic alarm.
SRFR 02	The end-user is be able to mark victims communicated by the firefighters, as well as their location and status publishing information with agents on the ground.
SRFR 03	The end-user must be able to activate the robot video camera (if it is present in the robot provided sensors list) to identify a potential victim and evaluate their state.
SRFR 04	The end-user must be able to define a new goal for a robot or a team of robots publishing a new area to scan.

Command and Control requirements

Table 3.2: Command and Control Functional Requirements

N.º	**Requirement**
C2FR 01	Change tactical mode of operations and communicate it to all the involved agents (Offensive, Defensive and Transitional).
C2FR 02	Classify the incident and fill an appropriate form to help him to make a risk assessment.
C2FR 03	Write a situational report and should be able to send it.
C2FR 04	Define initial coordinates of deployed robots on the ground and initiate their mission
C2FR 05	Record the developments of tactical situation: all tactical actions will be recorded in the command board to get an overview of the incident evolution
C2FR 06	Write/draw task information and communicate it to the agents.

Situation Awareness requirements

Table 3.3: Situation Awareness Functional Requirements

N.	**Requirement**
SAFR 01	Track robots on the ground
SAFR 02	Track firefighters on the ground
SAFR 03	Know the firefighter status
SAFR 04	Know the robot status (battery level, mission,...)
SAFR 05	Display local plant of activity
SAFR 06	Display temperature map
SAFR 07	Display gas map
SAFR 08	Know the location of hazardous materials
SAFR 09	Know the location of fire outbreaks, their state of evolution and the class of fire
SAFR 10	Record areas where searches were carried out
SAFR 11	Display images supplied by the robot
SAFR 12	Record existing damage on site
SAFR 13	Display incident time line

Self-Contained Breathing Apparatus requirements

Table 3.4: Self-Contained Breathing Apparatus Functional Requirements

N.º	**Requirement**
BAFR 01	Know the SCBA users and team composition
BAFR 02	Know the SCBA characteristics (capacity, level,..)
BAFR 03	Know the mission time of the SCBA users
BAFR 04	The system should know the level of the SCBA, the consumption rate and the distance from the exit to estimate the turn-around-time.
BAFR 05	Calculate the turn-around-time (TAT), display it on the screen and alert the SCBA Entry Control Officer and the firefighter when that time is reached.

Resources Management requirements

Table 3.5: Resources Management Functional Requirements

N.º	**Requirement**
PMFR 01	Know the resources deployed in action and in reserve, and their commitment within the mission.
PMFR 02	Know the communication plan and the assigned communication channels.
PMFR 03	Record the existing command structure on site.

User Warnings requirements

Table 3.6: User Warnings Functional Requirements

N.º	**Requirement**
UWFR 01	Status of all visual aids should be displayed to the controller
UWFR 02	Alarms and warnings to alert the end user of risk situations
UWFR 03	Alarms to warn the user: 1. Acoustic alarms 2. Visual alarms
UWFR 04	The user can disable alarms, but they will be activated again if no action is taken and the risk associated with alarm increases.

3.6.2 Non functional requirements

In a search and rescue scenario, all operations are controlled and coordinated by the commander of the rescue operations (Incident Commander (”Comandante das Operações de Socorro”) (COS)) that should be aware of the conditions of the incident and register it in the Tactical Situation (SItuação TÁCtica) (SITAC) form using adequate symbology [NOP09b].

The situation of the incident should be represented according to the forms, diagrams and symbols used by firefighters for easy reading and operation.

End User Operation

Table 3.7: End User Operation non Functional Requirements

N.º	**Requirement**
EUnFR 01	A sequence of actions should be designed to be logical from the perspective of the user, not from the perspective of computer processing or ease of programming
EUnFR 02	Interactive control logic should permit completion of a task with the minimum number of actions. However, this should not be to the detriment of situation awareness or consistency

Information Display

Information should be presented simply and in a well-organized. Ways to achieve simplicity include the following non functional requirements listed in Table 3.8:

Table 3.8: Information Display non Functional Requirements

N.º	**Requirement**
IDnFR 01	The screen should be orderly
IDnFR 02	Information should be presented in consistent, predictable locations
IDnFR 03	The language used should be plain and simple
IDnFR 04	Data items on a screen should be grouped on logical principles basis
IDnFR 05	Information shall be presented to a user in directly usable form; a user shall not have to decode or interpret data.
IDnFR 06	Data should be gathered logically to help the user in his tasks
IDnFR 07	Fonts used for texts should be legible with adequate size and contrast
IDnFR 08	The data needed for an interaction shall be displayed in a directly usable form
IDnFR 09	Information density should be minimized, in particular, for displays used for critical task sequences
IDnFR 10	Paging or scrolling can be used when the amount of information to display does not allow display on a single page
IDnFR 11	A system shall interrupt a user only when necessary to guide the user for a response, to provide essential feedback, or to inform the user of errors

Continued on next page

Table 3.8 – *Continued from previous page*

N.º	**Requirement**
IDnFR 12	The wording of displayed data and labels shall be chosen to reflect the user's point of view and shall correspond to the user's operational language
IDnFR 13	Words in the command language dialog shall reflect the user's point of view and shall correspond to the user's operational language
IDnFR 14	The response time of a system to a user action shall be appropriate to this type of action.
IDnFR 15	A maximum response time shall be determined for each type of action.
IDnFR 16	In case of a system failure during an interaction, the system should keep, as far as possible, the maximum of information in order not to have to start again the same interaction.
IDnFR 17	Colors, different levels of brightness, blinking colors shall be used to the following aims: 1. Show the agents status on the ground 2. Draw attention to emergencies 3. Identify critical situations 4. Improve readability and perception of the end user
IDnFR 18	The information should be presented in a way that non critical objects do not overlap with critical objects
IDnFR 19	To provide a clear awareness of the situation on the ground, the following information shall be displayed: 1. Different symbols for firefighters and robots 2. Target information (position, speed, identification, task, status) on labels and windows 3. Conflict alerts (visual and audible)
IDnFR 20	The user should be able to select the view he/she want, but the critical warnings of other views should appear to draw the attention to a new event.

3.7 Summary

This chapter defines the functional and non functional requirements of the application, providing an overview of the desired hardware characteristics and software interfaces to achieve the project goals. The choice of hardware technologies with touch capabilities, reliable open-source operating systems, standard software languages and consolidated frameworks, aims not only to ensure the system's robustness, but also to streamline the application development time and reduce production costs. Hereafter, the application interface requirements are defined, addressing some consolidated HMI techniques and methodologies, namely the Fitt's law, the Hick-Hymann law, the Tesler's law of complexity and the *poka-yoke* principle guiding the user in the system's operation, alerting for important events and preventing inadvertent user errors. A broader definition of context is also presented, fusing the traditional taxonomies of Schilit and Pascoe, combined with a HRI taxonomy proposed by Yanco and Drury [YD04] to support the overall situation awareness of the system, supporting the designed solutions presented in the next chapter.

Chapter 4

Designed Solution

4.1 Taxonomy of information flows

The information sharing between human and robotic teams and the Command Center (CC) is extremely important and it is required to use a common base for information validation and communication between the used platforms. This common base should be appropriate for data-interchange, language independent and human-readable, with two possible solutions to solve this issue, eXtensible Markup Language (XML) and JavaScript Object Notation (JSON).

XML is a subset of the Standard Generalized Markup Language (SGML) and is a universal data representation format. It is a user-defined hierarchical data format and is used to create user-defined markups for documents and encoding schemes, applied in object serialization for data transfer between applications.

JSON is designed to be a light-weight data-interchange format and easy for humans to understand. Although based on a subset of the JavaScript Programming Language, JSON is a text format that is language-independent, and is used for transmit data objects between applications, consisting of *attribute-value* pairs.

In an Urban Search and Rescue (USAR) scenario, fast and reliable communication is required and JSON claims a lightweight payload reducing bandwidth needs and allowing faster transfers, leading to lesser processing time [Cro06], but according to Lee [Lee13], given a document object, one can produce identical sized JSON and XML representations. Furthermore, this representations compress to nearly identical size which is an indicator that they contain approximately the same entropy or information content, so transferring these documents to a wide variety of devices takes effectively the same time per device [Lee13].

The ROSBridge protocol grants access to the underlying Robot Operating System (ROS) messages and services as serialized JSON objects, providing communication among the different agents in the system through a single web socket connection (Figure 2.7) enabling non-ROS users to publish and subscribe topics and call services as a true ROS node, consequently the messages are described in JSON format.

The analysis of the actions performed by agents on ground level, the relevant information to the performance of the mission and their interaction with the command center, leads to the identification of an initial set of messages for controlling and monitoring operations allowing collaborative context awareness. Messages are classified into five classes: *command & control*, *location*, *status*, *conditions* and *mapping*, according with their content, priority and function. The tree structure of the messages is presented in Figure 4.1.

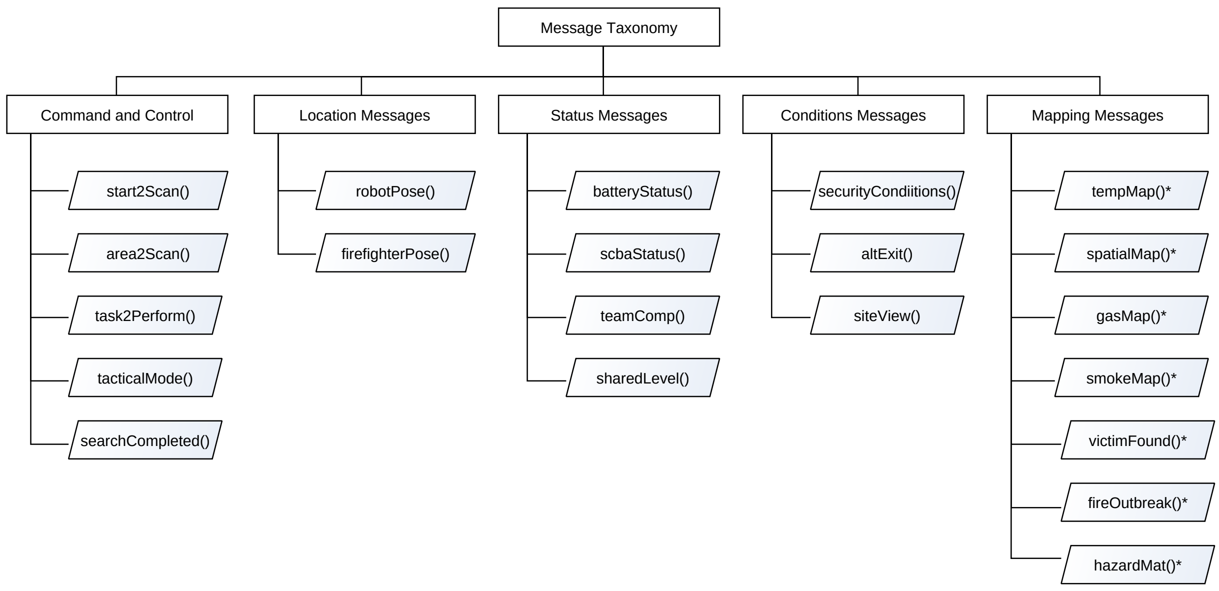


FIGURE 4.1: Tree structure of the messages between the CC, firefighters and robots (* indicates priority messages).

The messages are published/subscribed as ROS topics, with a defined *namespace* to identify the topic publisher and a field to indicate the receiver classes with their respective id. The CC will always have *id*="0" and the other agents *id's* will start with "1". If a message is sent to a robot or firefighter with *id*="0", the message should be broadcasted to all instances of the class. If an agent receive a ROS message with an $id \neq 0$ and different from its own *id* the message should be ignored.

A complete description of the messages is presented in Appendix B.

4.2 Design of the CC human-machine interface

The design of the CC human-machine interface allows the operator to define new incidents and navigate through previously registered occurrences, using a navigation bar [Figure 4.2]. This bar is located on the top of the screen, to meet the criteria of the Fitt's law, and uses typical graphic icons to illustrate the correspondent action, making operation easier for the end users.

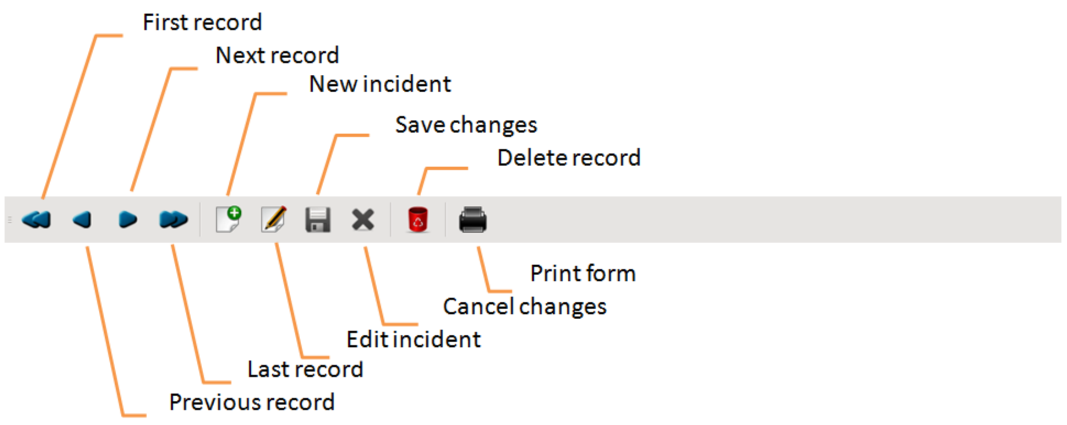


Figure 4.2: Incident navigation buttons.

Tactical actors can switch between the pages they are using, by selecting an option directly from the page selector at the bottom of the screen, or by turning across the pages using the arrow buttons on the left and right side of page selector bar [Figure 4.2].

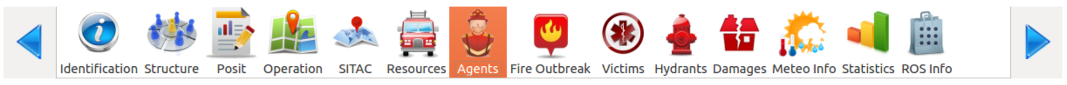


Figure 4.3: Screen selector.

The *Identification* screen [Figure 6.1(a)] is based on an National Authority for Civil Protection ("Autoridade Nacional de Protecção Civil") (ANPC) form, depending on the incident classification [Figure D.1]. This screen adopts the same layout and colors of the form to meet information display non-functional requirements.

To enhance situation awareness, the decision maker has a global overview of the structure of the incident and of the communication plan in the *Structure* screen [Figure 6.2(a)], organized in a tree structure, combining several forms already used in firefighters' operations [Figure D.6

and Figure D.10]. The entries in this screen are generated by actions taken in other screens in the application and depends on the incident phase level.

The incident commander and the sector commanders make situational reports whenever is required or in a thirty minutes time span, presented in the *POSIT* screen [Figure 6.2(b)], based on the ANPC form in Figure D.7. This screen layout provides also a summarized view of the entities and resources involved in the mission.

The definition of the Theatre of Operations ("Teatro de Operações") (TO) areas and sectors in the hot zone, as well as the deployment of resources on the ground and the adopted tactical mode are achieved in the Tactical Situation (SItuação TÁCtica) (SITAC) screen [Figure 6.1(b)], adopting the ANPC form [Figure D.4] and graphical symbology [Figure D.9] already used by the firefighters in their missions. This screen uses Googlemaps to overcome the constraint of lack of maps, faced by firefighters in many operations, providing a graphical tool for automating redundant tasks, since the operations in SITAC are stored in the database and the modified features has effects on other screens.

The remaining pages [Figure 6.5(a), Figure 6.5(b) and Figure 6.1] uses table view widgets to implement the forms used by the decision makers [Appendix D] addressing the *Resources*, *Agents*, *Fire outbreaks*, *Victims* and *Hydrants*, wherein the data are automatically filled by actions on other pages, or manually filled by the tactical actors to register a radio communication during the mission. The *damages* and *meteo info* screen are only filled manually by the operators.

The resources engaged in the incident, as well as their commitment to the mission and actual status are based in a proper form [Figure D.13] from ANPC and presented in Figure 6.5(a).

In the *Ros Info* screen [Figure 6.3(a)], the operators connects the CC with the Base Station (BS) through a WebSocket, to get a complete set of the available ROS messages published by the robots and firefighters on the field. These messages has a *namespace* to identify the publisher, and are presented to the user organized in a tree structure, on the left side of the screen. The tactical operator selects the desired message and subscribes it, to receive the data transmitted in that topic. All the subscribed topics are presented on a tree view on the right side of the screen. This approach allows the operator to see which topics are published and which are subscribed.

4.3 Interface with human and robotics first responders

The interface with human and robotic first responders in urban search and rescue activities represent a fundamental role in obtaining situation awareness. Whatever are the robots capabilities, the interface must provide the command center with sufficient information to make the correct decisions [MWH04]. High level in tasks autonomy of the robot relieves the commander of the operations to focus his attention in more high-level aspects.

The different requirements of the USAR mission determines different interfaces for different tasks, so it's critical that the commander of the operations keep a status awareness of the mission, so it's proposed an interface based on strategic games which have been developed in the last years for a better and intuitive interaction of the user and the scenario and their interveners, where it's possible to identify the agents, their location and status, as presented in Figure 6.3(b).

Communications with the BS and the ROS framework is achieved through the layout in Figure 6.3(a), where all the published topics and their publisher appear on the left and the subscribed topics appear on the right. The end user selects the desired topic to subscribe and move it to the subscribe window to start receiving information from the agent. Whenever a topic is subscribed, the publisher agent is added to the operational map, reflecting the agent status on the ground allowing the interaction between Command Center of Operations ("Centro de Comando de Operações") (CCO) and the agents.

Although robots try to find victims with their sensors, and the humans are commonly viewed as commanders, planners and managers, they are more noteworthy for their sensory feats [LW09] and could be helpful determining the victim condition, especially if the victim is trapped or unconscious, and only an arm or a leg is visible. So the interface is based on First-Person Shooter (FPS) video game which provides the primary source of surroundings awareness, as presented on the lower right corner of Figure 6.3(b).

Video and image transmission can overload the communication channel, therefore, the end user has the ability to connect/disconnect the video transmission.

4.4 Graphical SITAC

Google maps are a powerful Application Programming Interface (API) which facilitates the access to online maps with Global Positioning System (GPS) coordinates. These maps use a spherical Mercator projection based on WGS84. Firefighters use geographic coordinates in the Degree, Minutes, Seconds (DMS) format based on the global *datum* WGS84, which is the reference used by the GPS systems. Because earth is not a perfect sphere, *datums* are known as geographic shapes of the earth, which can be applied to maps so coordinate systems can still work perfectly, and they are classified as local *datums* and global *datums*. Conversion between *dms* (degree, minute, seconds) and *dd* (decimal degrees) coordinate formats is straightforward using Equation 4.1 and Equation 4.2.

$$dd = d + \frac{m}{60} + \frac{s}{3600} \tag{4.1}$$

$$dms = \begin{cases} d = trunc(dd) \\ m = trunc(|d| * 60)\ mod\ 60 \\ s = (|dd| * 3600)\ mod\ 60 \end{cases} \tag{4.2}$$

Using google maps, the CCO can construct the graphical SITAC with *drag&drop* operations with immediate knowledge of the geographic coordinates of resources deployed on the ground.

So every time a symbol is placed or dragged in the map, a *signal* is emitted, with the coordinates and description of the *JavaScript* object, to a *slot* in the application updating the database with the object data.

A similar feature is used, to update the graphical SITAC, with a *signal* emitted in the application and connected to a *JavaScript* function loaded in the WebView.

Another important aspect for the CCO and operators of the application is the calculation of distances between two *latitude/longitude* points. The calculations are based on a spherical model of the earth, ignoring the ellipsoidal effects. Distance calculations between two *latitude/longitude* points are based on the *haversine* (*'half-versed-sine'* $\frac{(1-cos\theta)}{2} = sin^2(\frac{\theta}{2})$) as described by Roger Sinnot [Sin84]:

$$a = sin^2(\frac{\Delta\varphi}{2}) + cos(\varphi_1) * cos(\varphi_2) * sin^2(\frac{\Delta\lambda}{2}) \tag{4.3}$$

Where φ is the *latitude*, λ is the *longitude*, $\Delta\varphi = \varphi_2 - \varphi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$.

The shortest distance between two points over the earth's surface is given by:

$$c = 2 * atan2(\sqrt{a}, \sqrt{1-a}) \tag{4.4}$$

$$d = R * c \tag{4.5}$$

Where $R = 6\ 371\ 137\ m$, is the earth's mean radius, and the angles are in radians.

Taking advantage of this feature, one can use the distance from one arbitrary common point chosen by the Incident Commander ("Comandante das Operações de Socorro") (COS), defined as origin *(0,0)*, and click on the entrance where the robots are deployed and thereby determine the relationship between the robot coordinate frame and map coordinate frame.

4.5 Communication within ROS

Class *RosMsg()* is responsible for handling all communications with the web socket. Once the connection is established, the class queries ROS for published topics and receives all messages available in the WebSocket in the *onMessage()* slot. This query is fired out every 5 seconds to detect new published topics or the loss of communication with an agent on the ground. As stated before, messages topics are received in JSON format and have to be parsed, to become understandable for the application and the end user. Due the heterogeneous structures of ROS messages, a recursive function *parseMsg()* is used to handle the topics structure. The parsed data is inserted into a model to be presented in the interface in a tree style.

4.6 Map transmission

Robots use multi-robot simultaneous localization and mapping (MRSLAM)[Mar13] to generate a coherent global map of the environment using a team of cooperative robots, transmitted later to the CCO whenever a network link is available. The generated map is an *occupancy-grid* representation where the value in each cell represents the probability of occupancy, from 0 (*unoccupied*) to 100 (*occupied*), and -1 (*unknown*). Transmission of an *occupancy-grid* over a wireless connection through a WebSocket proved to take a long time due the default size of the grid. To overcome this constraint, the map is compressed before transmission and decompressed after it is received. The data is compressed with the *zlib* library, using the *deflate* algorithm which is a combination of the *Lempel-Ziv (LZ77)* algorithm [ZL77] and *Huffman coding* [Huf52]. This algorithm is lossless which means the original data can be fully reconstructed.

The *LZ77* algorithm eliminates duplicate bytes by inserting a back-reference link to the previous location of the identical byte sequence, then the *Huffman coding* replaces the most frequently used symbols by shorter bit-sequence representations and the less used symbols with longer bit-sequence representations, completing the second stage compression.

Deflate is not the most efficient compression algorithm, but is implemented in a manner not covered by patents [Deu96], leading to its widespread use, with free implementations like the *zlib* library distributed under the Berkeley Software Distribution (BSD) license without advertising clause.

To test this feature and improve the transmission of maps between the BS and the CCO application, a ROS node was developed to subscribe the */map* or */Globalmap* topic, compress the map and publish the new topic */compressedMap*. CCO subscribe this topic and uncompresses the received map, showing it on the *Operational Map* screen.

```
    listener robot_1 robot_1/map
```

Code Excerpt 4.1: *Launch ROS node to compress and publish the map*

Comparative results of transmission time through the WebSocket of uncompressed and compressed map (672×460 occupancy grid), are presented in Figure 4.4:

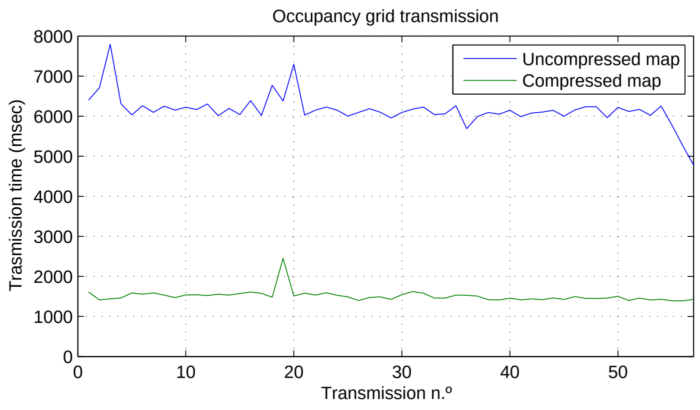


Figure 4.4: Transmission time of a map through the WebSocket.

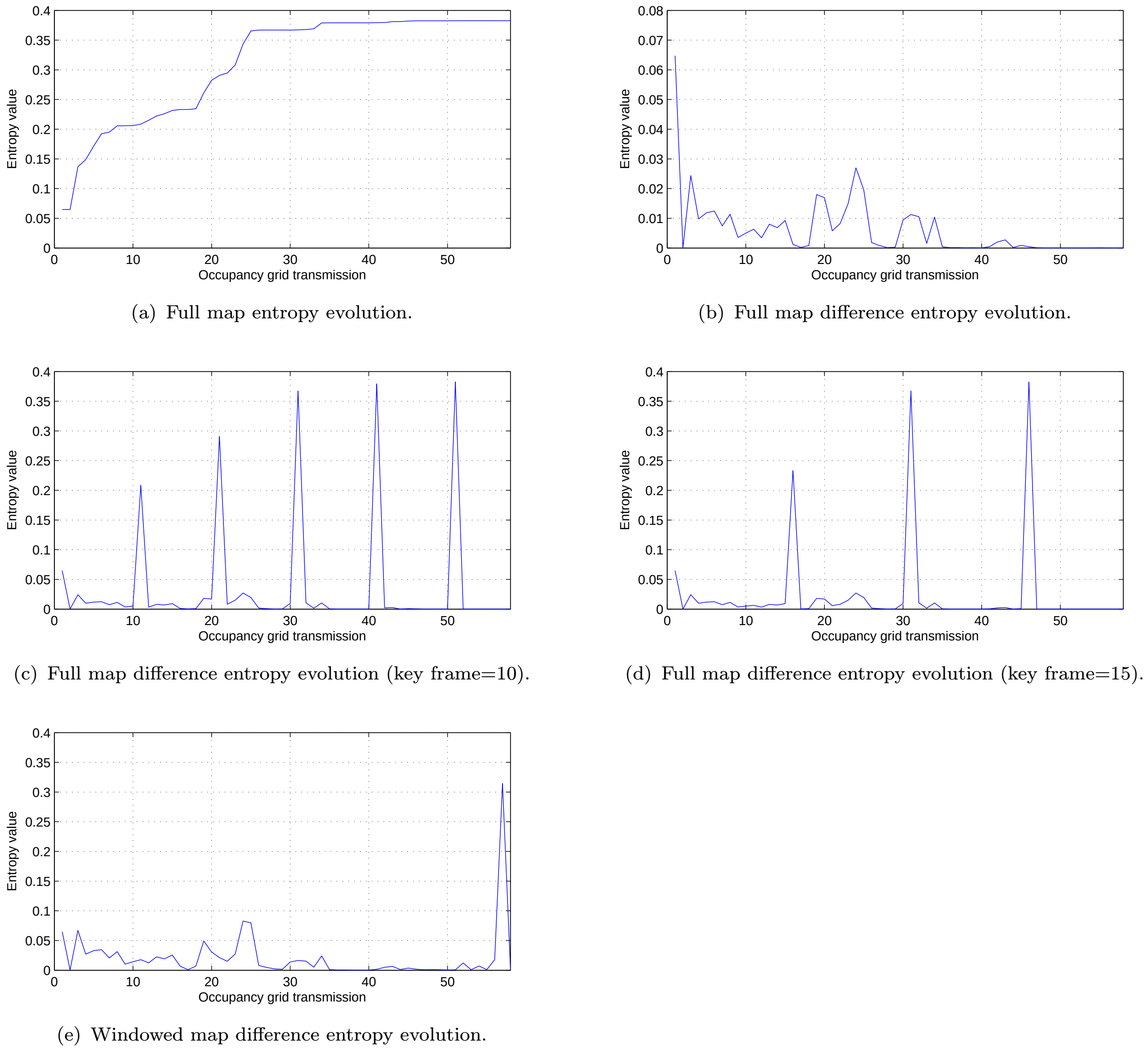
Shannon [Sha48] formulated the theory of data compression, stating that there is a limit to lossless data compression, called entropy rate, denoted by H [Equation 4.6].

$$H(X) = -\sum_{i=1}^{n} p_i * log_2 p_i \tag{4.6}$$

where p_i is the probability of the i-th value of the information source and $log_2 p_i$ is the information content. The values in the information source are statistically independent.

Over time, the *occupancy-grid* presents different sized information sources with different distributions, thus calculating the Shannon entropy for each information source, to quantify the expected value of information content to be transmitted through the WebSocket, one can expect a larger compressed file whenever the information is transmitted.

As the grid-based map become more accurate, the information content rises, leading to a higher entropy rate [Figure 4.5(a)] and consequently larger files to transmit through the WebSocket [Figure 4.6(a)] creating a bigger latency in the CCO application.



(a) Full map entropy evolution.

(b) Full map difference entropy evolution.

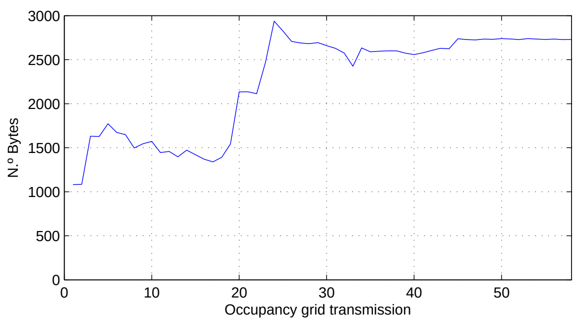
(c) Full map difference entropy evolution (key frame=10).

(d) Full map difference entropy evolution (key frame=15).

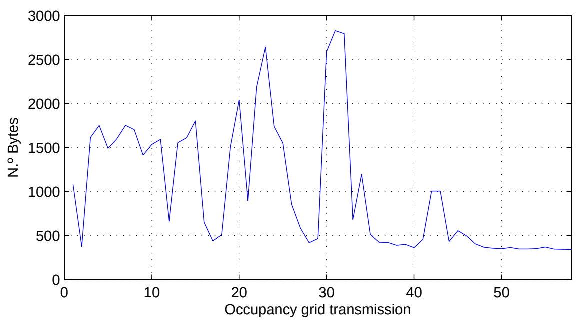
(e) Windowed map difference entropy evolution.

FIGURE 4.5: Shannon entropy measurements (672x460 occupancy grid).

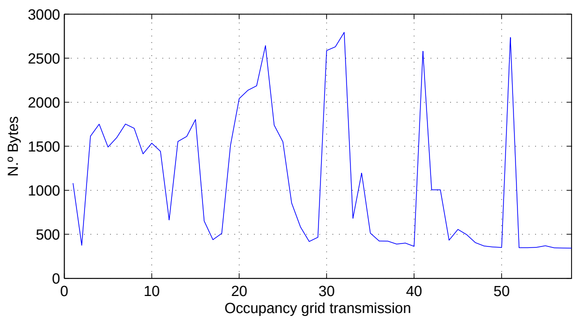
Different strategies where tested to improve the map transfer and reduce the *bottleneck* effect originated in the WebSocket. Calculating the differences between the last transmitted



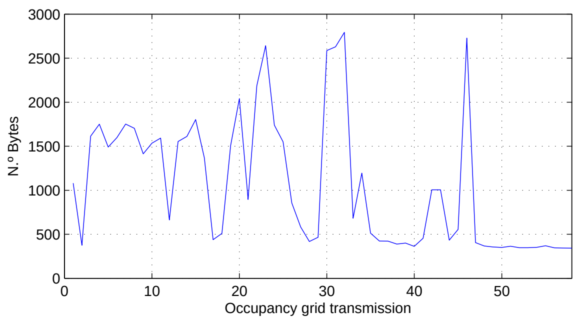
(a) Full map transmission (130 488 bytes).



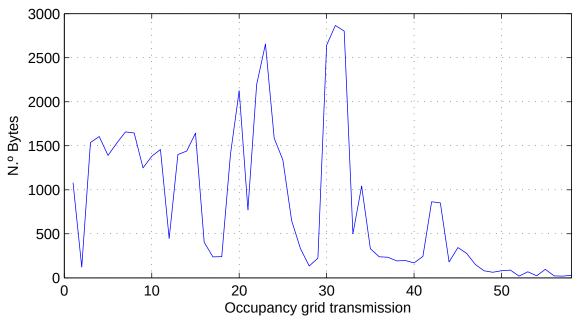
(b) Full map difference transmission (58 837 bytes).



(c) Full map difference w/ key frame (10) (64 224 bytes).



(d) Full map difference w/ key frame (15) (61 587 bytes).



(e) Window map difference transmission (48 567 bytes).

FIGURE 4.6: Transmitted bytes measurements (672x460 occupancy grid).

map and the new map, leads to an information source with lesser entropy [Figure 4.5(b)], and consequently a higher compression ratio, with a small number of bytes through the communication channel [Figure 4.6(b)].

Due to communication link failures over a time span, parts of the occupancy-grid fail to reach the CCO and the map will be incoherent. To overcome this constraint a key frame is established and in a communication time span, the entire map is transmitted filling eventual missing parts of the map. As expected, this approach leads to higher entropy values [Figure 4.5(c) and Figure 4.5(d)], compared with the simple map difference strategy, and consequently to a smaller compression ratio, increasing the number of bytes through the WebSocket [Figure 4.6(c) and Figure 4.6(d)], but enabling a more reliable map transmission.

Large *occupancy-grids* (e.g. 2048 × 2048 cell grid) cause an overhead in the communication channel, even with compressed data, so another algorithm was tested, where the map differences defines the window boundaries of the data to send to the CCO and only the relevant information is transmitted through the web socket. This information is added to the CCOs' map in the positions determined by the sender.

4.7 Robots features

Robots can communicate their pose in the map frame using a *geometry_msgs/Pose* message calculated by the robot or directly through a *tf/tfMessage* where all the information about the coordinate frames of the robots is available on any computer on the system, or yet using the node *robot_pose_publisher* which convert the transform into a pose message. So it is necessary to ensure that exists a valid transform between the */base_link* frame and the */map* frame. As stated before the published topics should contain a prefix to identify the robot even in the */tf* message (e.g. */robot_1/base_link*).

The transformation in free space between two coordinate frames is composed of a translation (*geometry_msgs/Vector3*) and a rotation (*geometry_msgs/Quaternion*) and a class *QQuaternion* is available in Qt, so the calculation of the robot pose from the transform is straightforward. Since:

$$P_{odom} = Q_{base\_link} * P_{base\_link} * Q^{-1}_{base\_link} \tag{4.7}$$

where P_{base_link} is the vector 3D transformed into the quaternion $0 + iP_x + jP_y + kP_z$, combined with the rotation quaternion. Taking advantage of the quaternion properties, one can determine the pose of the robot in the map frame from Equation 4.8.

$$P_{map} = (Q_{odom} * Q_{base\_link}) * P_{base\_link} * (Q_{odom} * Q_{base\_link})^{-1} \tag{4.8}$$

Finally, the robot pose is translated to the screen frame.

4.8 Firefighters features

The use of a mobile device equipped with sensors by firefighters, as used in the development of the thesis work *iBombeiro* [Pin13], integrates the agents in the system through the framework *ROSBridge* allowing them to publish and subscribe topics as a true ROS node. The status of the agent is communicated to the CCO making use of the *firefighterPose* message [Code excerpt B.14].

As is not foreseen communicate the firefighters pose, the end-user has to update the agent position with *drag&drop* operations and broadcasting it on a topic message, along with the undergoing assigned mission, to reinforce context awareness and the established knowledge common ground on all actors in the mission. However, the *firefighterPose* message informs the CCO about the situation of the firefighter with a string field indicating if he or she is standing, climbing, descending, crouched or crawling allowing the user to infer on surrounding environmental conditions. If the firefighters stand still over a time span larger than 30 seconds an alarm is fired to warn CCO and the nearby agents, or if two or more agents progress on the ground crouching or crawling for a long period, indicates an unsafe exit or area to avoid, so the end-user provides actions to determine the location of the hazard zone and broadcast it on a topic to the agents on the ground.

When the firefighter initiates an assigned mission with an Self-Contained Breathing Apparatus (SCBA), starting time is registered establishing the effective work autonomy for the firefighter - A_{eff}(min), depending on the device capacity - C_{device}(liters), the pressure gauge initial reading - P_{gauge}(bar) and the estimated average consumption of air by a firefighter during heavy work - 40 l/min, where the air reserve of the apparatus lasts approximately 10 minutes, thus securing the return of the firefighter [Gue05].

$$A_{eff} = \frac{C_{device} \times P_{gauge}}{40} - 10 \tag{4.9}$$

Therefore the system publishes a topic with a warning sign to the CCO and the firefighter when the mission time exceeds half the effective work autonomy, reinforcing the warning as time approaches the reserve time, finally broadcasts an alert message to the nearby agents with the last known position of the agent.

4.9 Environmental variables

In a close environment, fire characteristics, rising temperatures and the resulting toxic gases can endanger the life integrity of firefighters and victims, creating unpredictable and risky conditions under low visibility for the first responders. Robots equipped with sensors read the environment variables to enhance situation awareness for the decision makers and firefighters anticipating the use of the right tools to face hazard situations and prevent more serious incidents.

Using multi-sensor fusion information, as proposed in the thesis work *MRsensing* [Fer13], within Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) project, classification of several environmental readings determine the surrounding characteristics and publish them in a topic [Code excerpt B.25] along with the position of the event.

The subscribed topics information is registered in the database and the end-user can choose to display or not the referred information.

4.10 Gesture recognition

In the operational map class, the *occupancy-grids*, the robots pose and all the events associated with victims and fire-outbreaks detection, as well the environment variables are represented enhancing situation awareness. Gesture recognition capabilities are added to the application to enable the end user to communicate with the computer and interact in a more natural way.

Qt includes a framework for gesture programming with some specialized classes to implement standard gestures. Although new gestures can be implemented, only standard gestures are used in this application, Code excerpt 4.2 illustrates how to activate specific gestures in *Qt*.

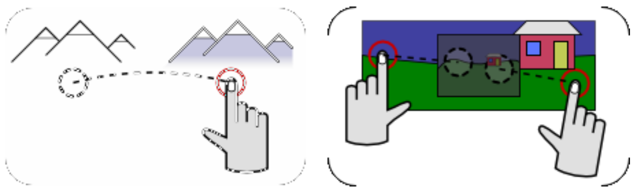

(a) Pan gesture. (b) Pinch.

Figure 4.7: Gesture recognition.

In this case, the application only uses the *Pan* [Figure 4.7(a)] and *Pinch* [Figure 4.7(b)] gestures to manipulate the map allowing the user to *rotate*, *zoom in* and *zoom out* the drawing area. One can achieve this reimplementing the general *event()* handler function [Code excerpt 4.3] and delegating gesture events to a user defined *gestureEvent()* function [Code excerpt 4.4]:

```
setAttribute(Qt::WA_AcceptTouchEvents);
grabGesture(Qt::PanGesture);
grabGesture(Qt::PinchGesture);
```

Code Excerpt 4.2: *Grab gestures in Qt.*

```
bool MapView::event(QEvent *event){
    if (event->type() == QEvent::Gesture) {
        gestureEvent(static_cast<QGestureEvent*>(event));
        return true;
    }
    return QWidget::event(event);
}
```

Code Excerpt 4.3: *general event() handler function.*

The proper response to gesture events is obtained examining the information contained in the specific *QGesture* object delivered in the *QGestureEvent*, according to the gesture state.

```
bool MapView::gestureEvent(QGestureEvent *event){
    if (QGesture *pan = event->gesture(Qt::PanGesture))
        panTriggered(static_cast<QPanGesture *>(pan));

    if (QGesture *pinch = event->gesture(Qt::PinchGesture))
        pinchTriggered(static_cast<QPinchGesture *>(pinch));

    return true;
}
```

Code Excerpt 4.4: *user defined gestureEvent() handler function.*

When gesture state is in *Qt::GestureUpdated*, the object acquires a large set of values from the input device and some abrupt changes can occur, due to read errors, originating jump responses of the system. Using the discrete implementation of a simple*RC* low-pass filter [Equation 4.10] the system response is smoothed, eliminating undesired variations in the *rotating* and *zooming* actions.

$$y_i = \alpha * x_i + (1 - \alpha) * y_{i-1} \tag{4.10}$$

where $\alpha = \frac{\Delta_T}{RC+\Delta_T}$, by definition, the smoothing factor is $0 \leq \alpha \leq 1$.

The user changes between screens in the interface with the *swipe* gesture, which is simulated using a *pan* gesture, allowing horizontal and vertical movements in changing pages [Code excerpt 4.5].

```
case Qt::GestureFinished:
    if (abs(gesture->lastOffset().x())>abs(gesture->lastOffset().y())){
        setVerticalMode(false);
        if (gesture->lastOffset().x()>0) slideInPrev();
        else slideInNext();
    } else {
        setVerticalMode(true);
        if (gesture->lastOffset().y()>0) slideInPrev();
        else slideInNext();
    }
```

Code Excerpt 4.5: *changing pages with pan Gesture handler function.*

4.11 Summary

This chapter presents the designed solutions used to fulfill the requirements stated in the previous chapter, proposing a message taxonomy, to ensure communication among the users of the system and maintaining situation-awareness. These messages are presented in ROS and JSON format to allow information interchange between the heterogeneous platforms.

Hereafter, the design of the screens adopt the layout of ANPC forms and symbology to improve the acceptance of end-users, leading to the implementation of a graphical SITAC, by integrating HTML, JavaScript with C++ language, thus providing a graphical tool to plan and control the operations on the ground.

In order to improve communications with ROS, when maps are transmitted through the WebSocket, some algorithms were tested, wherein the use of differences between maps produces smaller compressed data to be transmitted, thus reducing the transmission time and the application latency.

The developed features applied to robots, firefighters and environmental features are presented, where the information interchange is supported by the proposed message taxonomy and the database presented in the next chapter.

Chapter 5

Database Analysis

An Urban Search and Rescue (USAR) mission can involve a large number of resources and personnel, from different entities, and can attain a high degree of division complexity and communications channels can be overloaded by the information flow to report occurrences originated in the incident. Capability to register events is added to the Command Center of Operations ("Centro de Comando de Operações") (CCO) with a database to capture the structure of the mission and maintain historical data.

Due to the large number of entities involved, one can follow a top-down approach to define the conceptual data model using the *clustering concept* where collections of entities and relationships comprise higher-level objects, allowing to represent the entire database conceptual schema [Figure 5.1], where the entity clusters are depicted in yellow and maintain the same relationships between entities inside and outside the entity cluster, as occur between the same entities in the lower-level diagram. No attributes are presented in this higher-level diagram to provide a clear view of the major functionalities areas of the database.

From the database requirement analysis, six major groups and their relationships are identified:

- the incident itself;
- the deployed resources (teams, vehicles, robots);
- the personnel (decision makers, specialists, firefighters);
- the incident divisions (sectors, cells, sections, areas);
- the occurrences originated by the incident (victims, fire outbreaks, tactics);

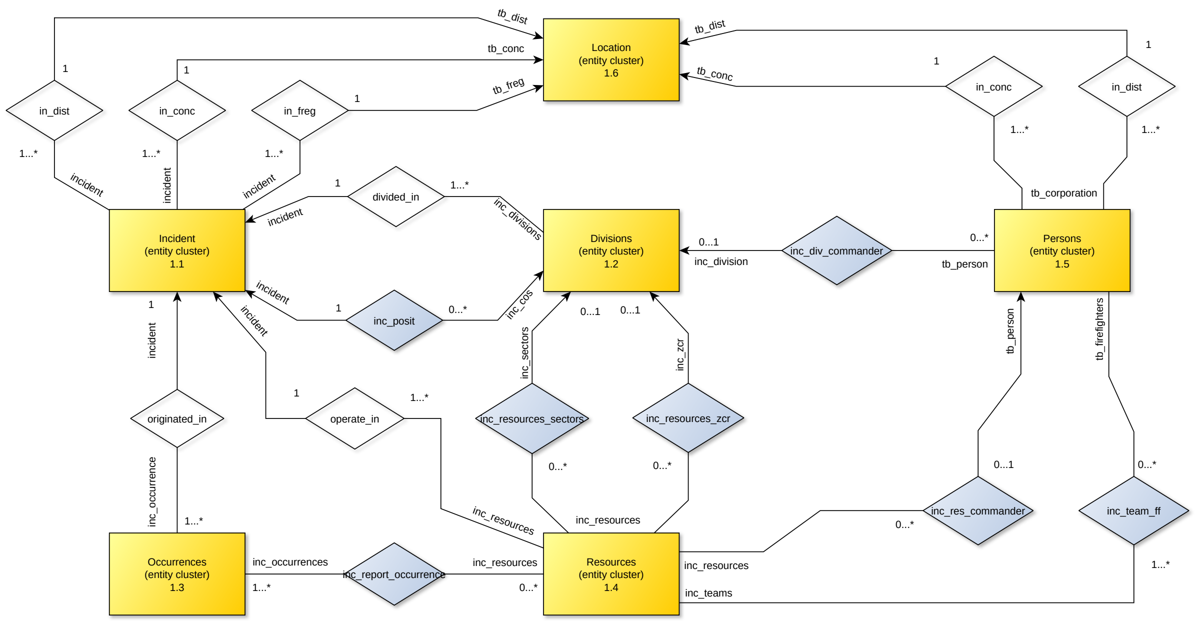


FIGURE 5.1: Database conceptual schema - *root entity cluster*.

- the location of the incident.

The incident occurs in a location classified by the Portuguese regional country division, characterized by three relationships with the location cluster

Each division of the incident is commanded by a person of a firefighters' corporation or an external entity and has a location in Theatre of Operations ("Teatro de Operações") (TO), changing over time depending on the incident evolution (*inc_div_commander*).

The resources working in the incident are located in a sector (*inc_sectors*) or in the concentration and reserve area (*inc_zcr*). The resources are commanded by a person, and eventually this commander can change over time, yielding a *1:M* relationship optional on both sides, therefore *inc_res_commander* holds the *tb_person* and *inc_resources primary keys*. The same rationale applies to the *all "many-to-many"* (*N:M*) relationships *inc_team_ff* and *inc_report_occurrence* regardless the involved entities are mandatory or optional. All actions are coordinated by the Incident Commander ("Comandante das Operações de Socorro") (COS) which have to communicate the *inc_posit* in a predefined time span.

5.1 Conceptual cluster models

5.1.1 Incident cluster

The incident cluster is expanded in Figure C.2 showing the associated entities and respective relationships and also the relationships with other clusters. The first entity involved is the "***incident***", and the kind of occurrence determines the used form. This table registers all data of the main event, time of occurrence, location and classification. Different topics are gathered depending on the incident classification [NOP09a], and the incident is a generalization of four types of occurrences (*forest fires*, *structural incidents* (Figure D.1), *hazard materials incidents* and *traffic accidents*), so depending on its complexity, a different structure is defined (*Divisions cluster*) and resources are mobilized and deployed on the ground (*Resources cluster*). The sequence of events is registered in the occurrence table (*Occurrences cluster*).

5.1.2 Incident divisions cluster

The structure of the incident is defined by its complexity, depending on the phase of the incident, and the number of resources deployed on the ground, so every incident is divided into sectors with operating resources, which can change over the incident time span, and an incident commander (*inc_cos*) to coordinate the operations, chosen by his rank and expertise, which can have a group of assistants (*inc_assistants*) as liaison agents with external entities. As the incident grows in complexity and number of resources, other operational and strategic units are settled (*inc_cells*) with an assigned commander, like the operational cell which is responsible to coordinate the incident sectors (*inc_sectors*), the planning cell, assisted by specialists (*inc_specialists*) depending on the nature of the incident, and the logistics cell which is responsible for the coordination of the concentration and reserve zone (*inc_zcr*).

The *inc_zcr* comprise several areas (*sgo_zcr*) depending upon the incident complexity and deployed resources.

5.1.3 Incident occurrences cluster

The incident occurrence cluster is expanded in Figure C.4 and the main entity *inc_occurrence* is a generalization of the different events transmitted by the operating resources, to register the maps transmitted by the robots (*inc_maps_table*), the encountered fire-outbreaks (*fire_outbreak*), the reported structural damages in the building (*damages*), the potential hazard environment in an area (*hazard_condition*) and the victims found and their state (*inc_victims*). Some occurrences can have a position or a pose stored in *inc_occurrence_pose.*

To enhance context-awareness of the incident commander, fire outbreaks are classified with a class (*tb_fire_class*) and a status (*tb_fire_status*), and the discovered victims are characterized with a state (*tb_victim_state*), a condition (e.g. "surface", "trapped", "entombed", "unknown") (*tb_victim_condition*), a situation (e.g. "aware", "semi-aware", "unconscious", "unknown") (*tb_victim_situation*), a type (e.g. "civilian", "firefighter")(*tb_victim_type*) and a nationality (*tb_nationalities*).

Although these latter entities have only one attribute, besides the added primary key, they are transformed into tables to support modifications in classification, if needed, determined by a NOP change, without altering the application .

5.1.4 Incident resources cluster

The incident resources cluster is expanded in Figure C.5 and the main entity *inc_resources* is a generalization of firefighter teams (*inc_teams*), vehicles (*inc_vehicles*) and robots (*inc_robots*).

Each firefighter's team is assigned to a vehicle, but a team can be replaced for another or the vehicle can fail and be replaced with by another, so this is represented by a *1:M* relationship, mandatory on the *one* side and optional on the *M* side, leading to a relationship (*inc_vehicle_team*). The remaining relationships are *1:M* mandatory on the *M* side so the *primary key* of the entity on the *"one"* side is an attribute of the entity on the *"many"* side.

The entities *tb_robot_class*, *tb_vehicle_type* and *tb_vehicle_group* define the characteristics of the vehicles and robots deployed on the ground.

5.1.5 Incident personnel cluster

Personnel in the TO comprise firefighters (*tb_firefighter*), members of the National Authority for Civil Protection ("Autoridade Nacional de Protecção Civil") (ANPC) (*tb_anpc*) and elements of other entities (*tb_others*) aggregated in the generalization entity *tb_person.* Each person is a member of an entity (*tb_entities*). Firefighters and ANPC members have an identification number which could be used as primary key, but actors who belong to other entities could not have one, also person can be a firefighter in an incident and an in another, leading to a key violation error in *tb_person*, so an *auto-incremented primary key* is defined to uniquely identify the actors attached to the incident. Each person represents a role in the incident, which can be a division commander (operations cell, logistics cell, incident commander, supporting officer, . . .) or a resource commander (team commander).

In hazard environments, firefighters use a Self-Contained Breathing Apparatus (SCBA) and their lives depend on the breath apparatus characteristics and autonomy, these devices are registered in a table (*inc_scba*) and affected to a firefighter. It is possible to a firefighter to use different breath apparatus, and one device be used by two or more agents (*inc_ff_scba*).

Firefighters have a hierarchical structure depending on a rank (*tb_firefighter_rank*), which can be used to define the chain of command of the incident or of a team.

5.1.6 Incident location cluster

The incident location cluster is expanded in Figure C.7 and describes the Portuguese administrative country division. These divisions comprise districts (*tb_dist*), where each district divided in municipalities (*tb_conc*) and each municipality divided in civil parishes (*tb_freg*). Although the territory was redefined under statistical regions and sub regions known as Nomenclature of Territorial Units for Statistics (NUTS), during European integration, they do not have legal status in law, so fire fighting and ANPC missions' locations are still classified using the former administrative division of the territory.

5.2 Data modeling

The conceptual data model is implemented using Structured Query Language (SQL), by transforming the Entity-Relationship Diagrams (ERDs) into three different kinds of tables addressing to the following rules:

- Tables with the same information content as the original entity, when the binary relationships are *many-to-many* (M:M), *one-to-many* (1:M) on the *one* side; or *one-to-one* (1:1) on either side - the only exception to this rule is when the relationship is 1:1 mandatory on both sides, and in this case the two entities are transformed in only one table, and entities with any ternary or higher-degree relationship or generalization hierarchy.

- Tables with foreign key from the mandatory side of the relationship, when the binary relationships are *one-to-many* (1:M) for the entity on the M side; or *one-to-one* (1:1) for the entity that is not mandatory.
- Tables derived from relationship containing the foreign keys of all the entities in the relationship, when the binary relationships are *many-to-many* (M:M), relationships binary recursive and all relationships that are ternary or higher degree.

Whilst there are some dependent *weak* entities and relationships, unique identifiers are defined with an auto incremented field to overcome this constrain and streamline the programming work with the *model-view-delegate* in *Qt*. The data model for each table in the system is presented in Appendix C.

5.3 Summary

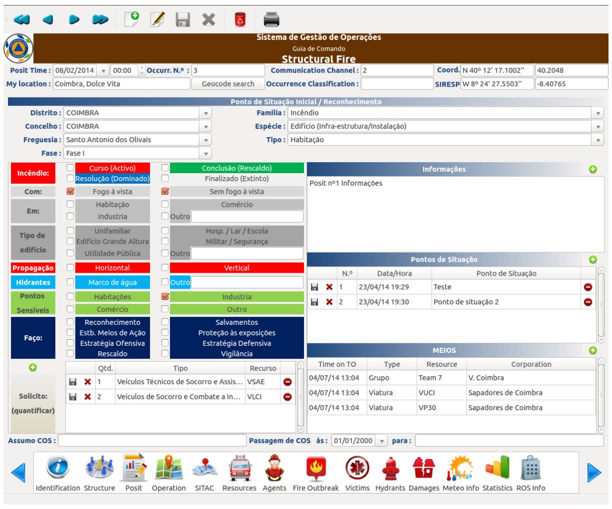
This chapter introduces the database analysis to register all relevant events of the incident, introducing a clustering technique to aggregate the entities within the same functional areas and their correspondent relationships, leading to the conceptual data model. Hereafter, the transformation rules to convert the conceptual data model into SQL are presented, leading to the CCO database where all the events of the incident are registered.

Chapter 6

Results

6.1 Incident simulation and CCO operation

When first responders arrive at the incident scene, after an emergency call, the incident commander size-up the incident and assess the risks, filling out the *Identification Screen* [Figure 6.1(a)].



Sistema de Gestão de Operações
Guia de Comando
Structural Fire

Posit Time : 08/02/2014 00:00 Occurr. N.º : 3 Communication Channel : 2 Coord. N 40º 12' 17.1002" 40.2048
My location : Coimbra, Dolce Vita Geocode search Occurrence Classification : SIRESP W 8º 24' 27.5503" -8.40765

Ponto de Situação Inicial / Reconhecimento

Distrito : COIMBRA
Concelho : COIMBRA
Freguesia : Santo Antonio dos Olivais
Fase : Fase I
Familia : Incêndio
Espécie : Edifício (Infra-estrutura/Instalação)
Tipo : Habitação

Incêndio:	Curso (Activo) / Resolução (Dominado)	Conclusão (Rescaldo) / Finalizado (Extinto)
Com:	[✓ Fogo à vista]	[✓ Sem fogo à vista]
Em:	Habitação / Industria	Comércio / Outro
Tipo de edifício	Unifamiliar / Edifício Grande Altura / Utilidade Pública	Hosp. / Lar / Escola / Militar / Segurança / Outro
Propagação	Horizontal	Vertical
Hidrantes	Marco de água	Outro
Pontos Sensíveis	Habitações / Comércio	[✓ Industria] / Outro
Faço:	Reconhecimento / Estb. Meios de Ação / Estratégia Ofensiva / Rescaldo	Salvamentos / Proteção às exposições / Estratégia Defensiva / Vigilância

Solicito: (quantificar)

Qtd.	Tipo	Recurso
1	Veículos Técnicos de Socorro e Assis…	VSAE
2	Veículos de Socorro e Combate a In…	VLCI

Informações

Posit nº1 Informações

Pontos de Situação

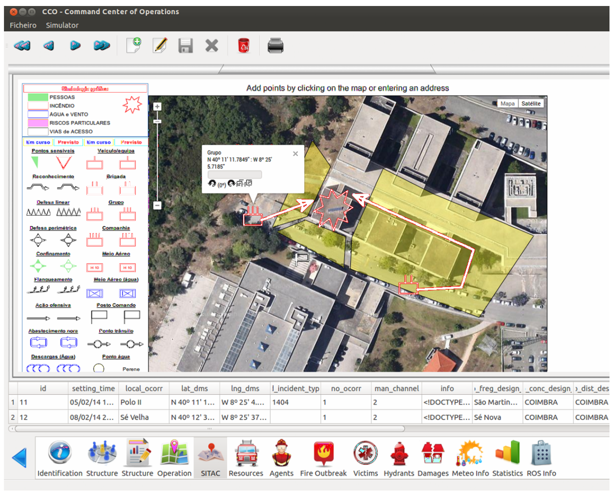
N.º	Data/Hora	Ponto de Situação
1	23/04/14 19:29	Teste
2	23/04/14 19:30	Ponto de situação 2

MEIOS

Time on TO	Type	Resource	Corporation
04/07/14 13:04	Grupo	Team 7	V. Coimbra
04/07/14 13:04	Viatura	VUCI	Sapadores de Coimbra
04/07/14 13:04	Viatura	VP30	Sapadores de Coimbra

Assumo COS : Passagem de COS às : 01/01/2000 para :

(a) Incident identification window.



CCO - Command Center of Operations
Ficheiro Simulator
Add points by clicking on the map or entering an address

	id	setting_time	local_ocorr	lat_dms	lng_dms	l_incident_typ	no_ocorr	man_channel	info	freg_design_	_conc_design_	dist_des
1	11	05/02/14 1…	Polo II	N 40º 11' 1…	W 8º 25' 4….	1404	1	2	<!DOCTYPE…	São Martin…	COIMBRA	COIMBRA
2	12	08/02/14 2…	Sé Velha	N 40º 12' 3…	W 8º 25' 37…		1	2	<!DOCTYPE…	Sé Nova	COIMBRA	COIMBRA

(b) Graphical SITAC.

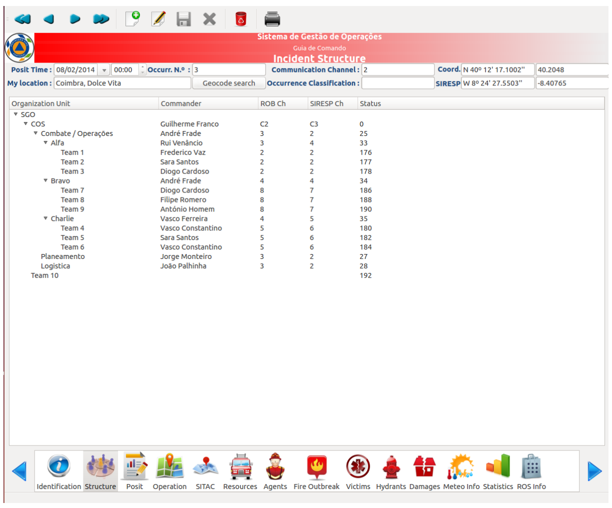
FIGURE 6.1: Human Machine Interface of the CC.

On the screen, in Figure 6.1(b), three sectors are defined (in yellow) and the resources are deployed using *drag&drop* operations providing their geographic coordinates location and the assigned tactical mode (offensive attack - arrow) to the tactical user, changing the *Structure*, *POSIT* and *Resources* screens.

The defined locations and deployed resources are stored in the database [Figure 6.5(a)] and the information necessary to characterize the context is transmitted to the correspondent agents, minimizing one of the DOODA's loop introduced delay, by affecting other screens, thus eliminating some redundancy.

As stated before, the incident structure is modified when the incident phase changes, therefore when the user changes the incident phase, the database will reflect the structure of the incident with all the resources involved and the communication plan with the assigned channels [Figure 6.2(a)].

Situation reports from commanders are stored in the database, providing the decision maker with the current situations within the incident [Figure 6.2(b)], providing also the summarized view of the entities and resources involved in the mission, defined in other screens, enhancing overall situation-awareness.

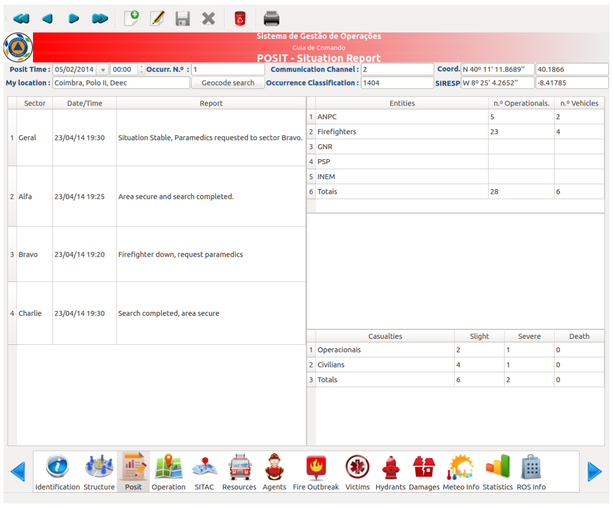


Posit Time : 08/02/2014 00:00 Occurr. N.º : 3 Communication Channel : 2 Coord. N 40º 12' 17.1002'' 40.2048

My location : Coimbra, Dolce Vita Occurrence Classification : SIRESP W 8º 24' 27.5503'' -8.40765

Organization Unit	Commander	ROB Ch	SIRESP Ch	Status
SGO				
COS	Guilherme Franco	C2	C3	0
Combate / Operações	André Frade	3	2	25
Alfa	Rui Venâncio	3	4	33
Team 1	Frederico Vaz	2	2	176
Team 2	Sara Santos	2	2	177
Team 3	Diogo Cardoso	2	2	178
Bravo	André Frade	4	4	34
Team 7	Diogo Cardoso	8	7	186
Team 8	Filipe Romero	8	7	188
Team 9	António Homem	8	7	190
Charlie	Vasco Ferreira	4	5	35
Team 4	Vasco Constantino	5	6	180
Team 5	Sara Santos	5	6	182
Team 6	Vasco Constantino	5	6	184
Planeamento	Jorge Monteiro	3	2	27
Logistica	João Palhinha	3	2	28
Team 10				192

(a) Structure layout and communications plan.



Posit Time : 05/02/2014 00:00 Occurr. N.º : 1 Communication Channel : 2 Coord. N 40º 11' 11.8689'' 40.1866

My location : Coimbra, Polo II, Deec Occurrence Classification : 1404 SIRESP W 8º 25' 4.2652'' -8.41785

	Sector	Date/Time	Report
1	Geral	23/04/14 19:30	Situation Stable, Paramedics requested to sector Bravo.
2	Alfa	23/04/14 19:25	Area secure and search completed.
3	Bravo	23/04/14 19:20	Firefighter down, request paramedics
4	Charlie	23/04/14 19:30	Search completed, area secure

	Entities	n.º Operationals.	n.º Vehicles
1	ANPC	5	2
2	Firefighters	23	4
3	GNR		
4	PSP		
5	INEM		
6	Totais	28	6

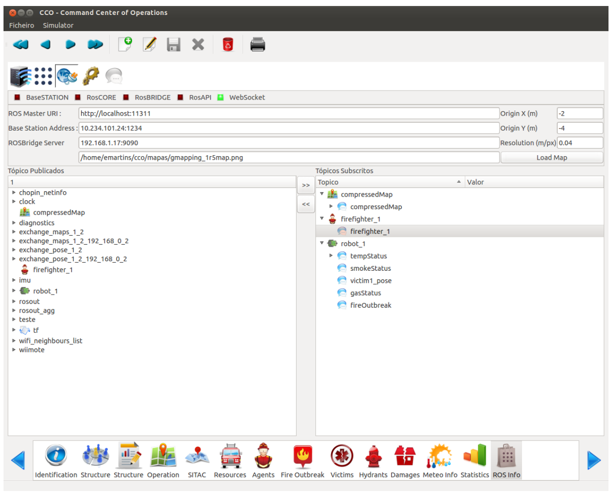
	Casualties	Slight	Severe	Death
1	Operacionais	2	1	0
2	Civilians	4	1	0
3	Totals	6	2	0

(b) Posit layout.

Figure 6.2: Incident structure, Posit and Communications plan layout.

The screen in Figure 6.3(a) allows to establish the connection between the Base Station (BS) and Robot Operating System (ROS), receiving data from the agents on field, displaying the operational map [Figure 6.3(b)] and incident features, while the tactical actor operates the application [Figure 6.4(a) and Figure 6.4(b)].

During the course of the mission, several occurrences should be discovered and reported to the Command Center of Operations ("Centro de Comando de Operações") (CCO), which are grouped according with their category, presented in proper layouts and registered in the database, alerting and prompting the incident commander to take proper actions.


CCO - Command Center of Operations

Ficheiro Simulator

BaseSTATION RosCORE RosBRIDGE RosAPI WebSocket

ROS Master URI : http://localhost:11311 Origin X (m) -2

Base Station Address : 10.234.101.24:1234 Origin Y (m) -4

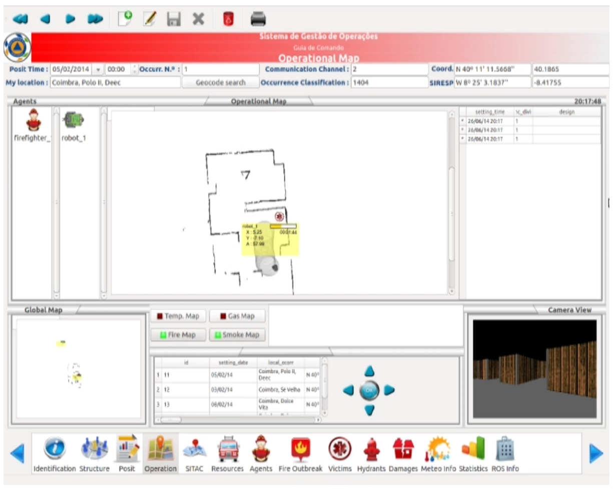
ROSBridge Server 192.168.1.17:9090 Resolution (m/px) 0.04

/home/emartins/cco/mapas/gmapping_1r5map.png Load Map

Tópico Publicados: chopin_netinfo, clock, compressedMap, diagnostics, exchange_maps_1_2, exchange_maps_1_2_192_168_0_2, exchange_pose_1_2, exchange_pose_1_2_192_168_0_2, firefighter_1, imu, robot_1, rosout, rosout_agg, teste, tf, wifi_neighbours_list, wiimote

Tópicos Subscritos — Topico / Valor: compressedMap (compressedMap), firefighter_1 (firefighter_1), robot_1 (tempStatus, smokeStatus, victim1_pose, gasStatus, fireOutbreak)

(a) ROS layout.


Sistema de Gestão de Operações

Guia de Comando

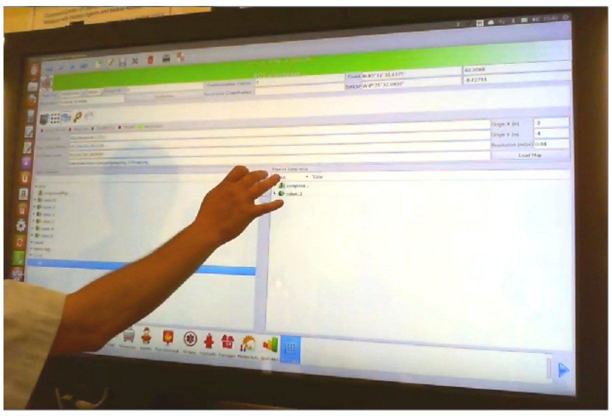
Operational Map

Posit Time : 05/02/2014 00:00 Occurr. N.º : 1 Communication Channel : 2 Coord. N 40º 11' 11.5668" 40.1865

My location : Coimbra, Polo II, Deec Geocode search Occurrence Classification : 1404 SIRESP W 8º 25' 3.1837" -8.41755

(b) Operational Map.

FIGURE 6.3: Human Robot Interaction of the CC.


(a) Subscribing ROS topics.

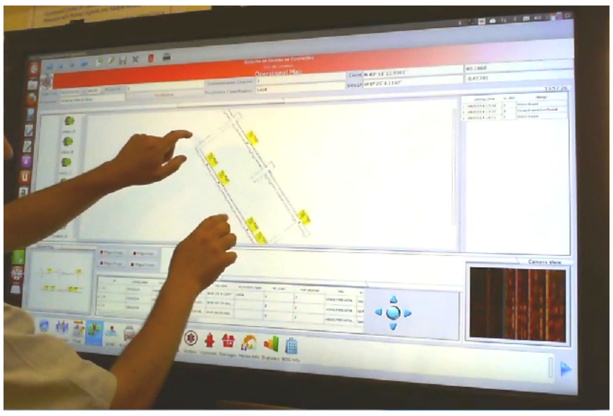
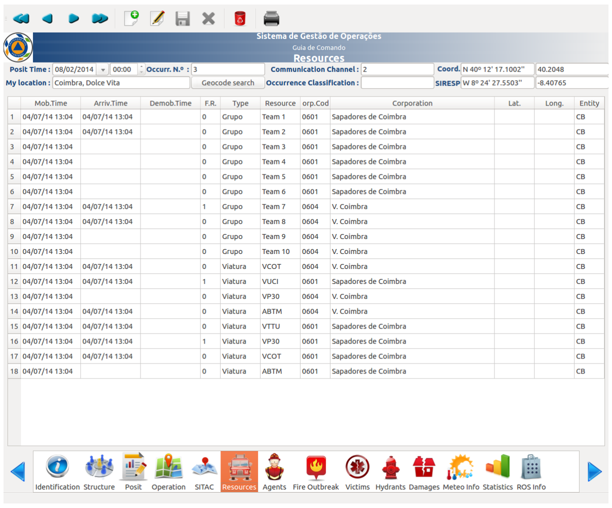

(b) Rotate and Zoom operations.

FIGURE 6.4: Touch screen Operation.

These automated tasks simplify the command and control of various features, such as entry control of Self-Contained Breathing Apparatus (SCBA) teams, by defining a time-based turning point for of the SCBA users [Figure 6.5(b)], where a coloured control bar indicates the SCBA status and the firefighter commitment within the mission, or creating an entry on victims map, holding the evolution and state of the discovered victims incident [Figure 6.1].

6.2 Summary

This chapter presents the result of the several interaction design techniques applied in the definition of the Human Machine Interface (HMI) for the CCO application, presented in

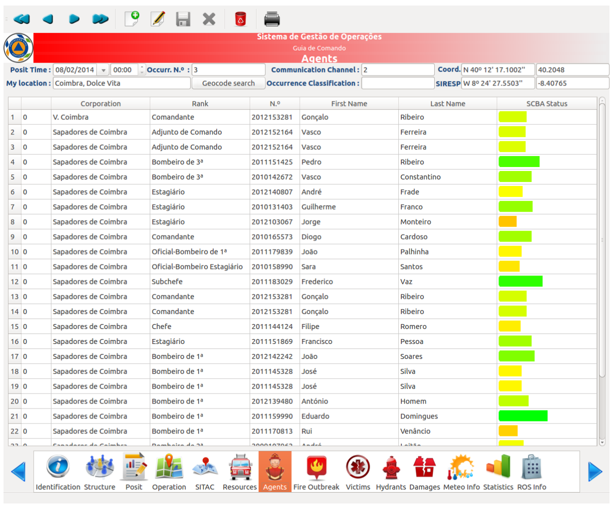


Posit Time : 08/02/2014 00:00 Occurr. N.º : 3 Communication Channel : 2 Coord. N 40º 12' 17.1002" 40.2048

My location : Coimbra, Dolce Vita Occurrence Classification : SIRESP W 8º 24' 27.5503" -8.40765

	Mob.Time	Arriv.Time	Demob.Time	F.R.	Type	Resource	orp.Cod	Corporation	Lat.	Long.	Entity
1	04/07/14 13:04	04/07/14 13:04		0	Grupo	Team 1	0601	Sapadores de Coimbra			CB
2	04/07/14 13:04	04/07/14 13:04		0	Grupo	Team 2	0601	Sapadores de Coimbra			CB
3	04/07/14 13:04			0	Grupo	Team 3	0601	Sapadores de Coimbra			CB
4	04/07/14 13:04			0	Grupo	Team 4	0601	Sapadores de Coimbra			CB
5	04/07/14 13:04			0	Grupo	Team 5	0601	Sapadores de Coimbra			CB
6	04/07/14 13:04			0	Grupo	Team 6	0601	Sapadores de Coimbra			CB
7	04/07/14 13:04	04/07/14 13:04		1	Grupo	Team 7	0604	V. Coimbra			CB
8	04/07/14 13:04	04/07/14 13:04		0	Grupo	Team 8	0604	V. Coimbra			CB
9	04/07/14 13:04			0	Grupo	Team 9	0604	V. Coimbra			CB
10	04/07/14 13:04			0	Grupo	Team 10	0604	V. Coimbra			CB
11	04/07/14 13:04	04/07/14 13:04		0	Viatura	VCOT	0604	V. Coimbra			CB
12	04/07/14 13:04	04/07/14 13:04		1	Viatura	VUCI	0601	Sapadores de Coimbra			CB
13	04/07/14 13:04			0	Viatura	VP30	0604	V. Coimbra			CB
14	04/07/14 13:04	04/07/14 13:04		0	Viatura	ABTM	0604	V. Coimbra			CB
15	04/07/14 13:04	04/07/14 13:04		0	Viatura	VTTU	0601	Sapadores de Coimbra			CB
16	04/07/14 13:04	04/07/14 13:04		1	Viatura	VP30	0601	Sapadores de Coimbra			CB
17	04/07/14 13:04	04/07/14 13:04		0	Viatura	VCOT	0601	Sapadores de Coimbra			CB
18	04/07/14 13:04			0	Viatura	ABTM	0601	Sapadores de Coimbra			CB

(a) Resources.



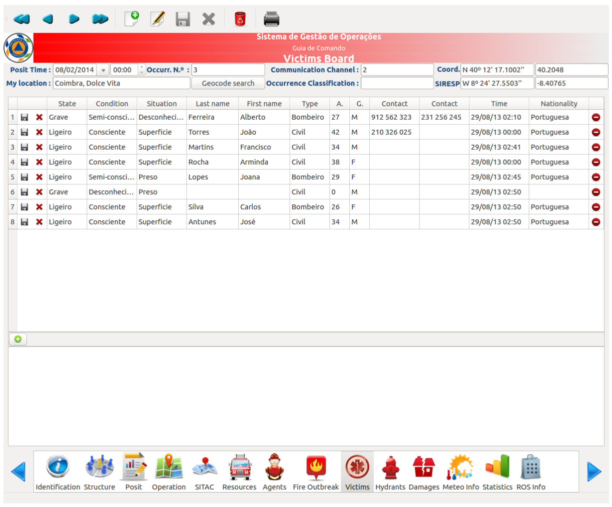
Posit Time : 08/02/2014 00:00 Occurr. N.º : 3 Communication Channel : 2 Coord. N 40º 12' 17.1002" 40.2048

My location : Coimbra, Dolce Vita Occurrence Classification : SIRESP W 8º 24' 27.5503" -8.40765

		Corporation	Rank	N.º	First Name	Last Name	SCBA Status
1	0	V. Coimbra	Comandante	2012153281	Gonçalo	Ribeiro	🟨
2	0	Sapadores de Coimbra	Adjunto de Comando	2012152164	Vasco	Ferreira	🟨
3	0	Sapadores de Coimbra	Adjunto de Comando	2012152164	Vasco	Ferreira	🟨
4	0	Sapadores de Coimbra	Bombeiro de 3ª	2011151425	Pedro	Ribeiro	🟩
5	0	Sapadores de Coimbra	Bombeiro de 3ª	2010142672	Vasco	Constantino	🟩
6	0	Sapadores de Coimbra	Estagiário	2012140807	André	Frade	🟨
7	0	Sapadores de Coimbra	Estagiário	2010131403	Guilherme	Franco	🟩
8	0	Sapadores de Coimbra	Estagiário	2012103067	Jorge	Monteiro	🟧
9	0	Sapadores de Coimbra	Comandante	2010165573	Diogo	Cardoso	🟩
10	0	Sapadores de Coimbra	Oficial-Bombeiro de 1ª	2011179839	João	Palhinha	🟨
11	0	Sapadores de Coimbra	Oficial-Bombeiro Estagiário	2010158990	Sara	Santos	🟨
12	0	Sapadores de Coimbra	Subchefe	2011183029	Frederico	Vaz	🟩
13	0	Sapadores de Coimbra	Comandante	2012153281	Gonçalo	Ribeiro	🟨
14	0	Sapadores de Coimbra	Comandante	2012153281	Gonçalo	Ribeiro	🟨
15	0	Sapadores de Coimbra	Chefe	2011144124	Filipe	Romero	🟨
16	0	Sapadores de Coimbra	Estagiário	2011151869	Francisco	Pessoa	🟩
17	0	Sapadores de Coimbra	Bombeiro de 1ª	2012142242	João	Soares	🟩
18	0	Sapadores de Coimbra	Bombeiro de 1ª	2011145328	José	Silva	🟨
19	0	Sapadores de Coimbra	Bombeiro de 1ª	2011145328	José	Silva	🟨
20	0	Sapadores de Coimbra	Bombeiro de 1ª	2012139480	António	Homem	🟨
21	0	Sapadores de Coimbra	Bombeiro de 1ª	2011159990	Eduardo	Domingues	🟩
22	0	Sapadores de Coimbra	Bombeiro de 1ª	2011170813	Rui	Venâncio	🟧

(b) Agents.

Figure 6.5: Resources and Agents layout



Posit Time : 08/02/2014 00:00 Occurr. N.º : 3 Communication Channel : 2 Coord. N 40º 12' 17.1002" 40.2048

My location : Coimbra, Dolce Vita Occurrence Classification : SIRESP W 8º 24' 27.5503" -8.40765

			State	Condition	Situation	Last name	First name	Type	A.	G.	Contact	Contact	Time	Nationality	
1		✗	Grave	Semi-consci...	Desconheci...	Ferreira	Alberto	Bombeiro	27	M	912 562 323	231 256 245	29/08/13 02:10	Portuguesa	⊖
2		✗	Ligeiro	Consciente	Superficie	Torres	João	Civil	42	M	210 326 025		29/08/13 00:00	Portuguesa	⊖
3		✗	Ligeiro	Consciente	Superficie	Martins	Francisco	Civil	34	M			29/08/13 02:41	Portuguesa	⊖
4		✗	Ligeiro	Consciente	Superficie	Rocha	Arminda	Civil	38	F			29/08/13 00:00	Portuguesa	⊖
5		✗	Ligeiro	Semi-consci...	Preso	Lopes	Joana	Bombeiro	29	F			29/08/13 02:45	Portuguesa	⊖
6		✗	Grave	Desconheci...	Preso			Civil	0	M			29/08/13 02:50		⊖
7		✗	Ligeiro	Consciente	Superficie	Silva	Carlos	Bombeiro	26	F			29/08/13 02:50	Portuguesa	⊖
8		✗	Ligeiro	Consciente	Superficie	Antunes	José	Civil	34	M			29/08/13 02:50	Portuguesa	⊖

Figure 6.6: Victims layout

several screens, aiming to enhance the overall situation-awareness and achieve the desired level of automation, promoting a better acceptance of the system by the end-users. The application records and monitors information coming from agents on the ground, producing operational warnings when critical situations are detected. Thus, minimizing the factors that can deteriorate situation-awareness, such as, *attentional-tunnelling*, *data overload*, *workload*, *anxiety*, *fatigue*, *misplaced salience*, *errant mental models* and *out-of-the-loop syndrome*, allowing the decision maker to be focused in analyzing data and making decisions.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This dissertation aimed the development of the Human Machine Interface (HMI) for the Command Center of Operations (”Centro de Comando de Operações”) (CCO) of Urban Search and Rescue (USAR) missions, exploring decentralized and collaborative techniques of collecting data to enhance Situation Awareness (SA), thus aiding decision making at the top-level command of the mission structure.

The system interface followed interaction design principles to facilitate the operation and ensure users' engagement. Were also taken into account context-awareness requirements in order to prioritize the relevant information for decision makers in the CCO, and Human-Robot Interaction (HRI) requirements to provide a common ground for the CCO-human-robot awareness. This was achieved with a messages taxonomy proposal, which provides a common base for data interchange among the operating actors on the ground.

The system implementation, in controlled laboratory environment, provides functionality results demonstrated by the interface's ability to represent the maps generated by the robots and status features transmitted by the robots and firefighters.

The integration of different technologies and operating environments, connected with a Web-Socket introduces a relevant delay constraint due to the dimensions of the generated map. Therefore, a solution with a combination of reduced map window differences and consequent compression with the *zlib* library was tested reducing the number of bytes received and application latency during maps transmission, although improvements were not those expected, mainly due to orientation alterations of the maps, leading to increased regions of interest to be transmitted.

7.2 Future Work

To meet the requirements that guide the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) project and provide a wide and integrated context-awareness becomes of major relevance to know the position of the firefighters on the ground, supported by the availability of multiple indoor positioning technologies and techniques and taken advantage of link quality parameter with the connected neighbors made available by the Mobile ad hoc Network (MANET) developed in the project.

Although mobile internet connections and Googlemaps presents improved quality and reliability, based on redundant systems and servers, one must consider a system failure and an alternative way to draw the Tactical Situation (SItuação TÁCtica) (SITAC) map on the screen, either by loading a previously saved map or allowing the user to draw a freehand map.

Aside the use of video cameras, one should consider the use of thermal cameras and/or other group of sensors with the ability to read and interpret classic fire behavior warnings, providing an augmented context-awareness useful to save firefighters lives.

Appendix A

Use Cases and System Models

A.1 Use-case diagrams

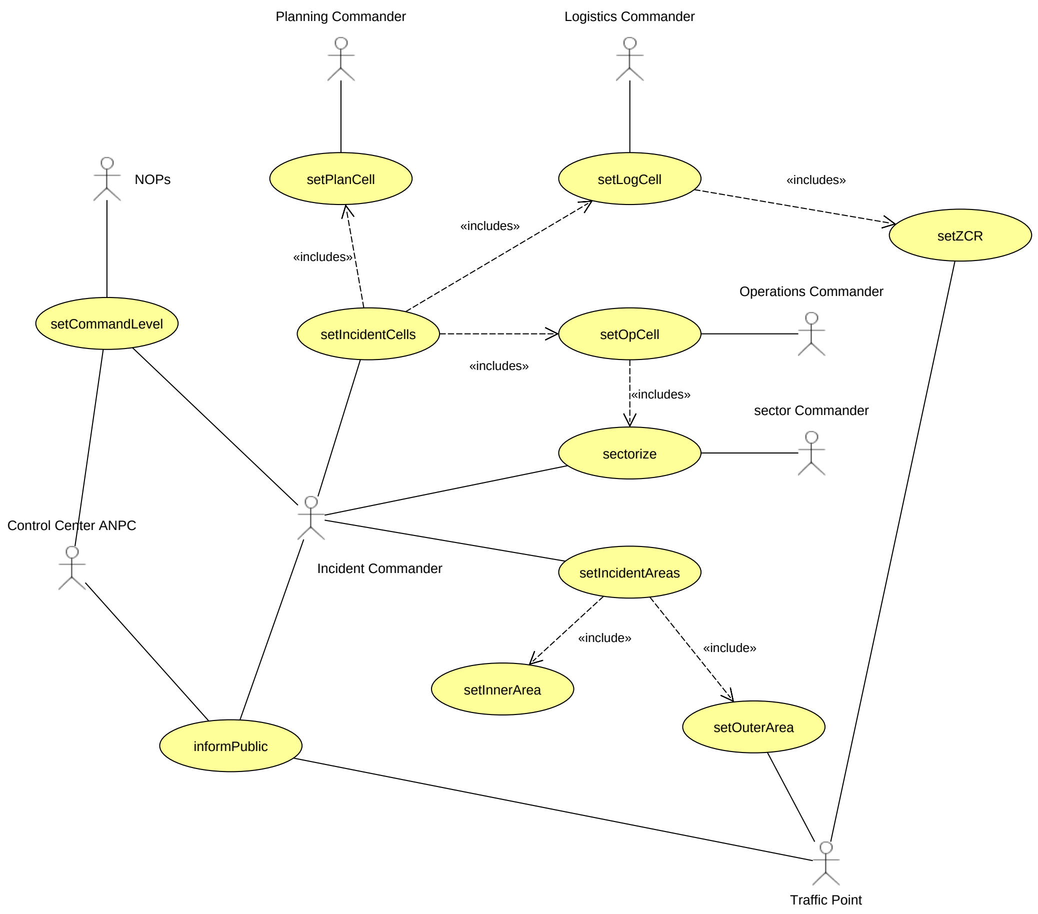


FIGURE A.1: Theatre of operations organization use-cases.

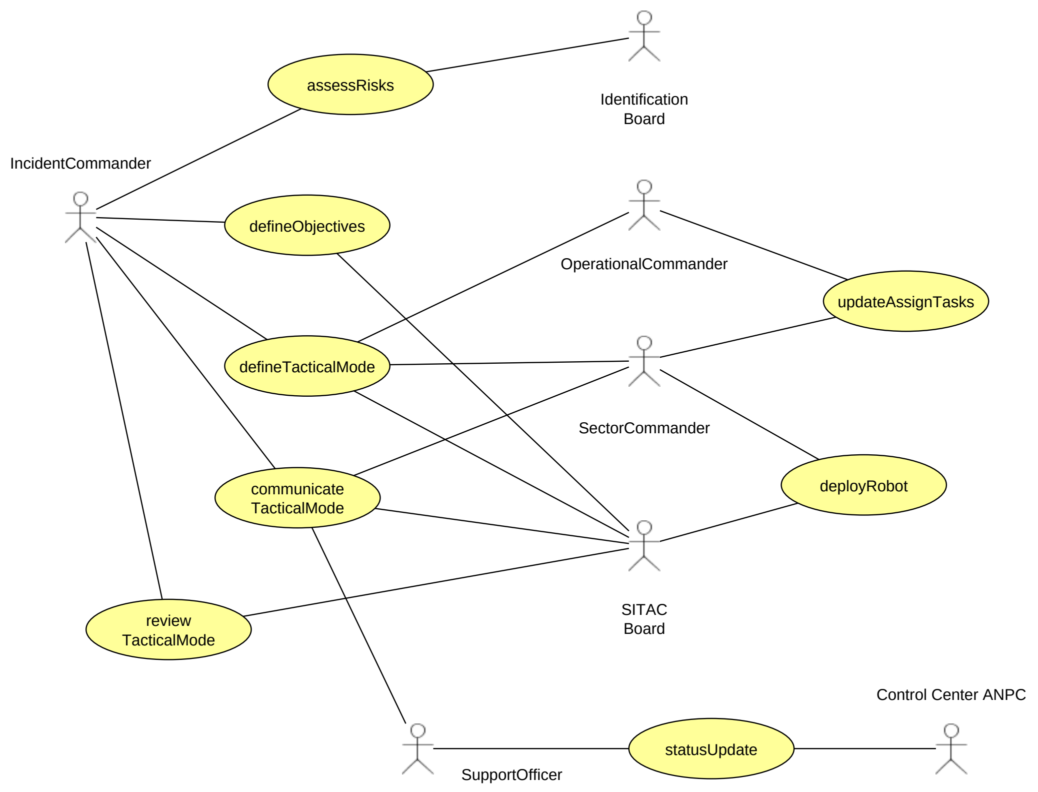


FIGURE A.2: Tactical Mode and Risk Assessment use-cases.

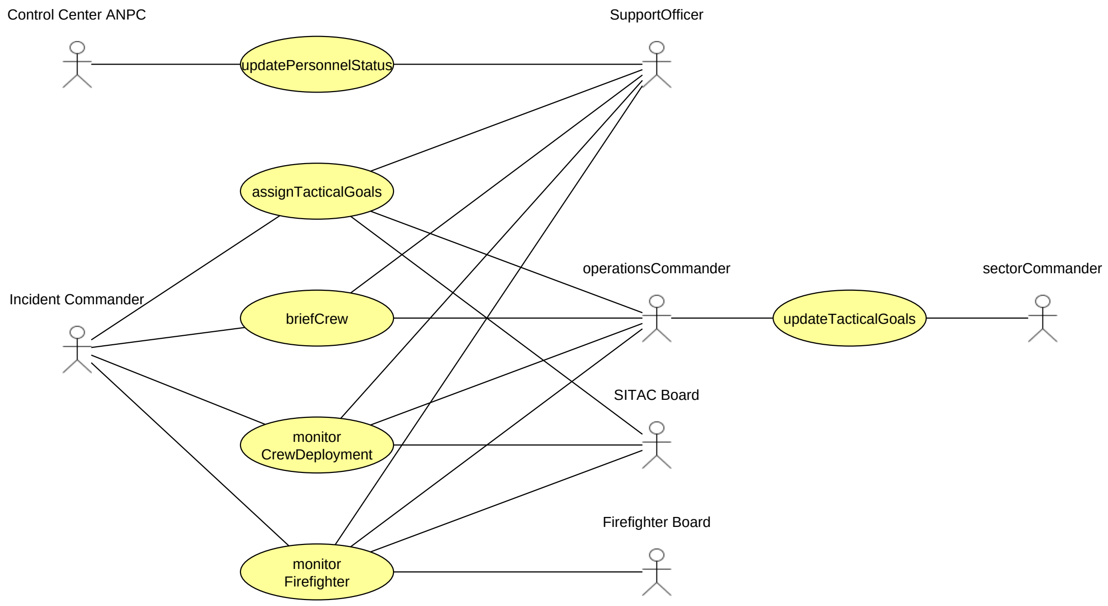


FIGURE A.3: Personnel Management use-cases.

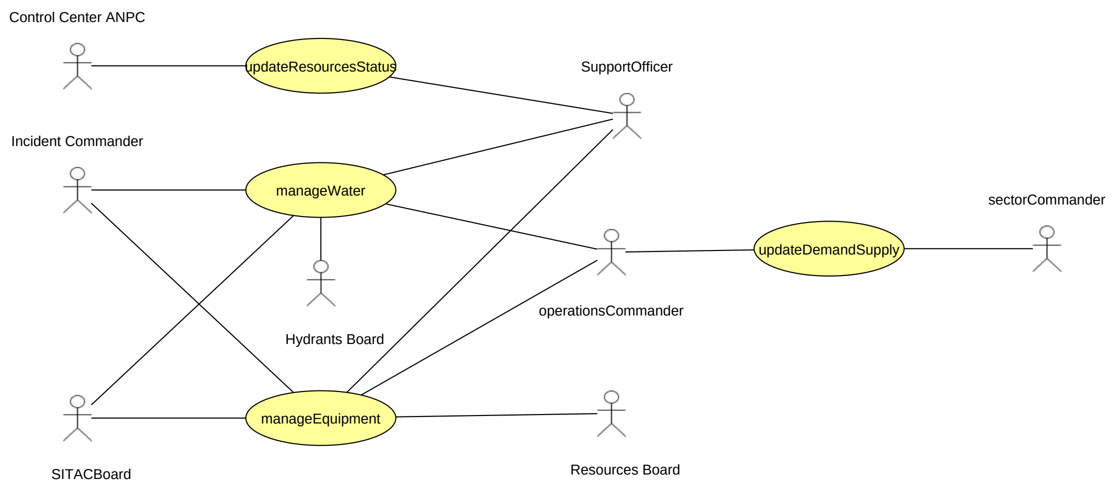


FIGURE A.4: Resources Management use-cases.

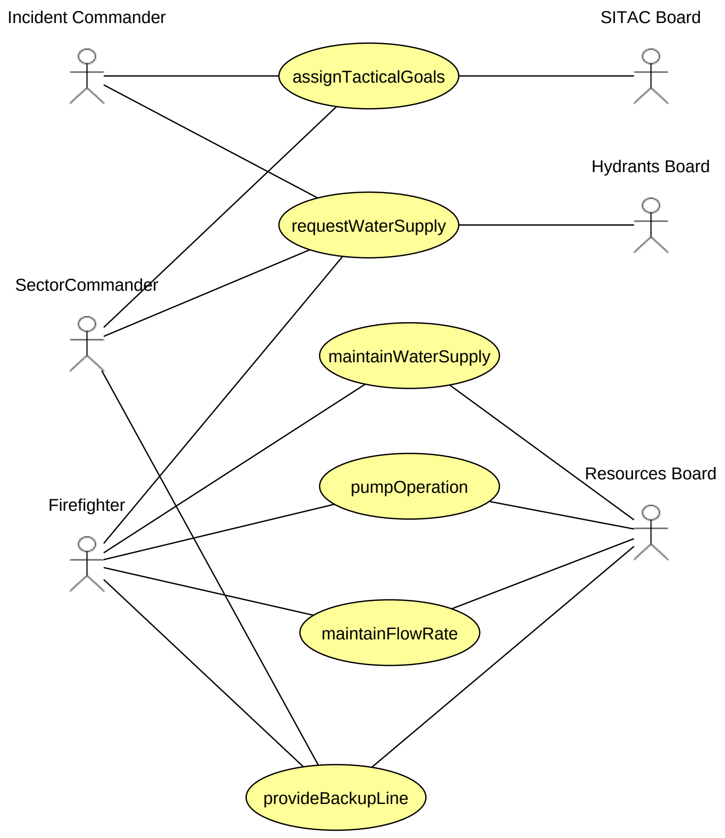


FIGURE A.5: Manage Water supply use-cases..

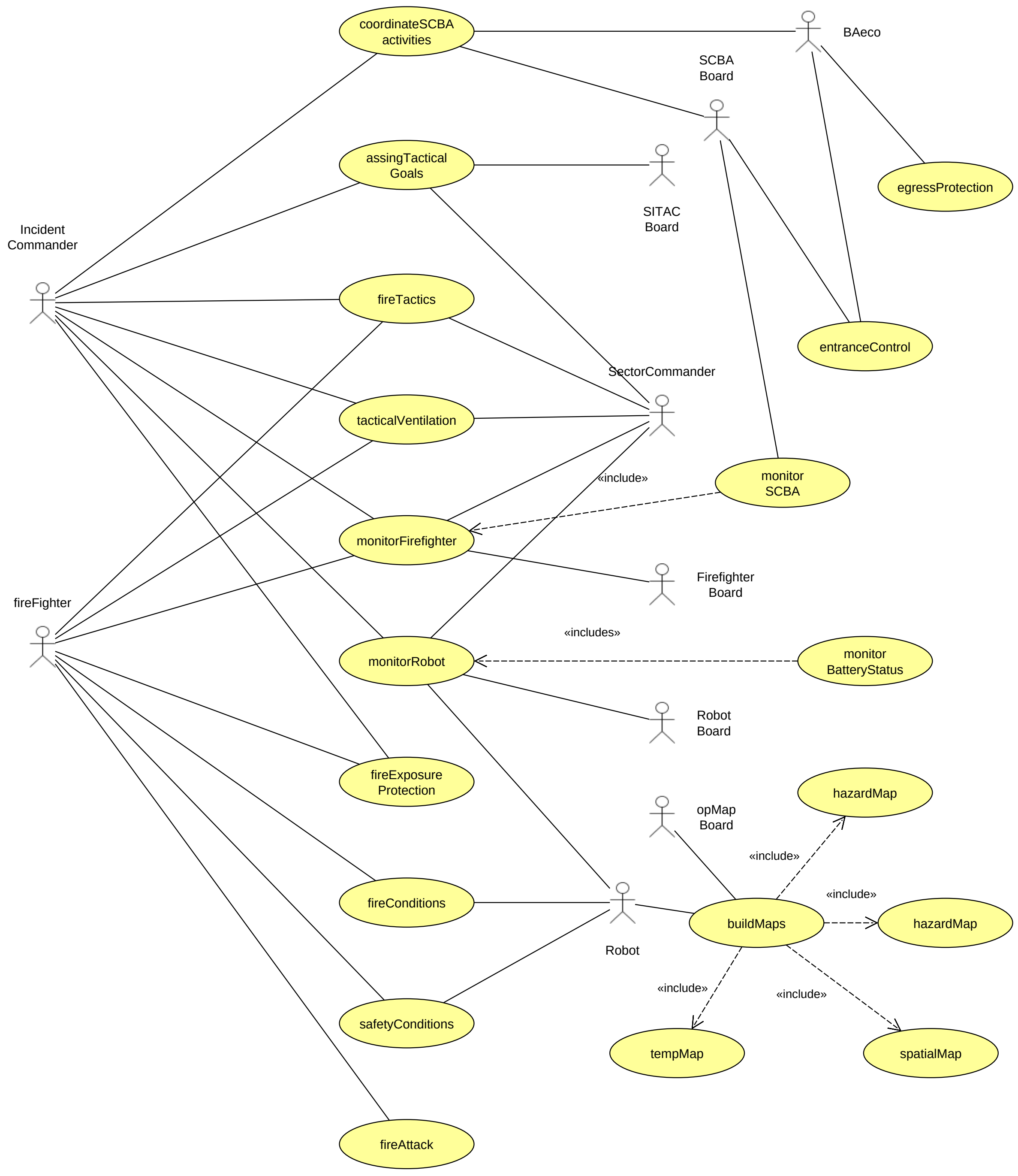


Figure A.6: Fire attack use-cases.

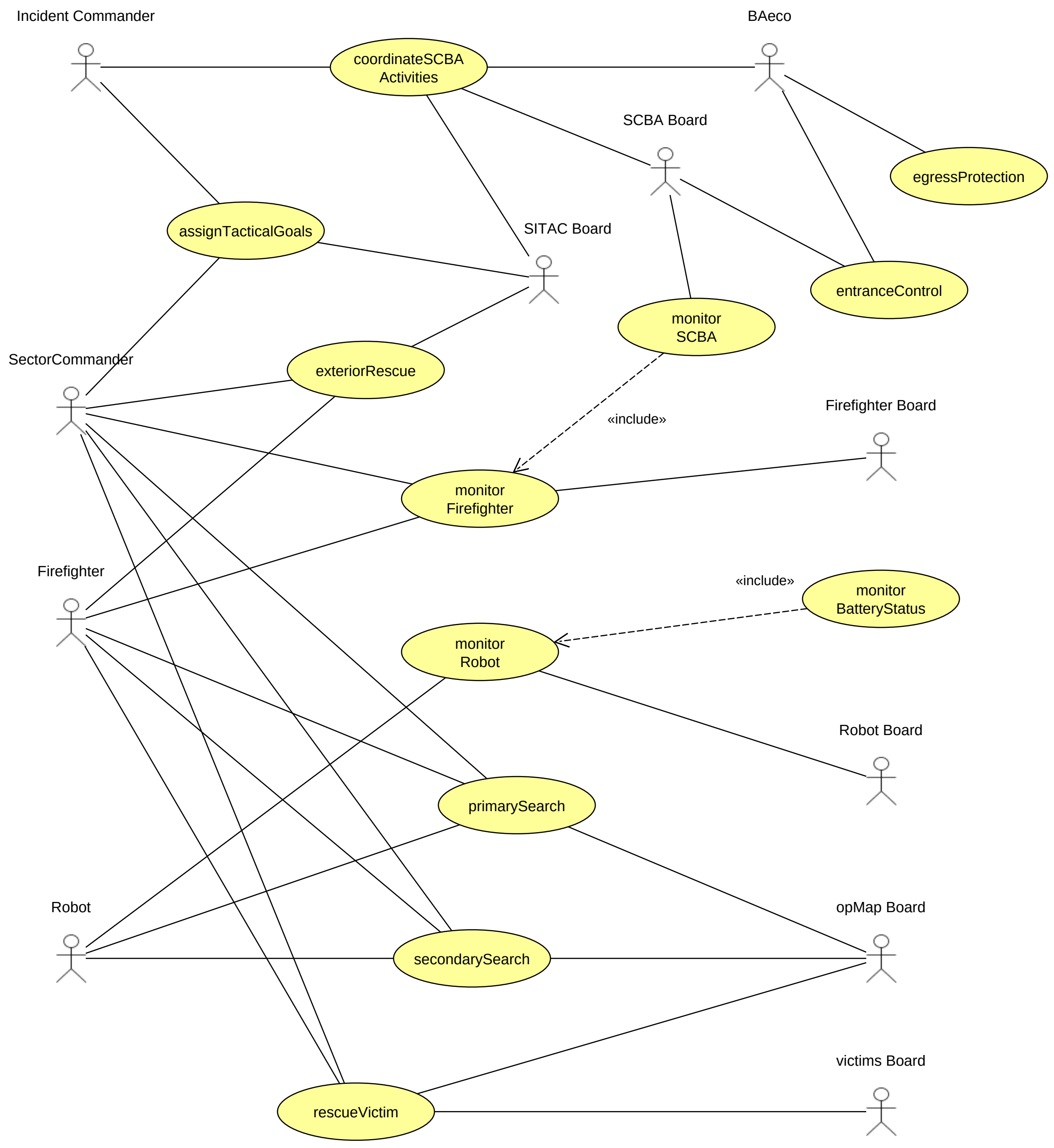


FIGURE A.7: Search and Rescue use-cases.

A.2 Class diagrams

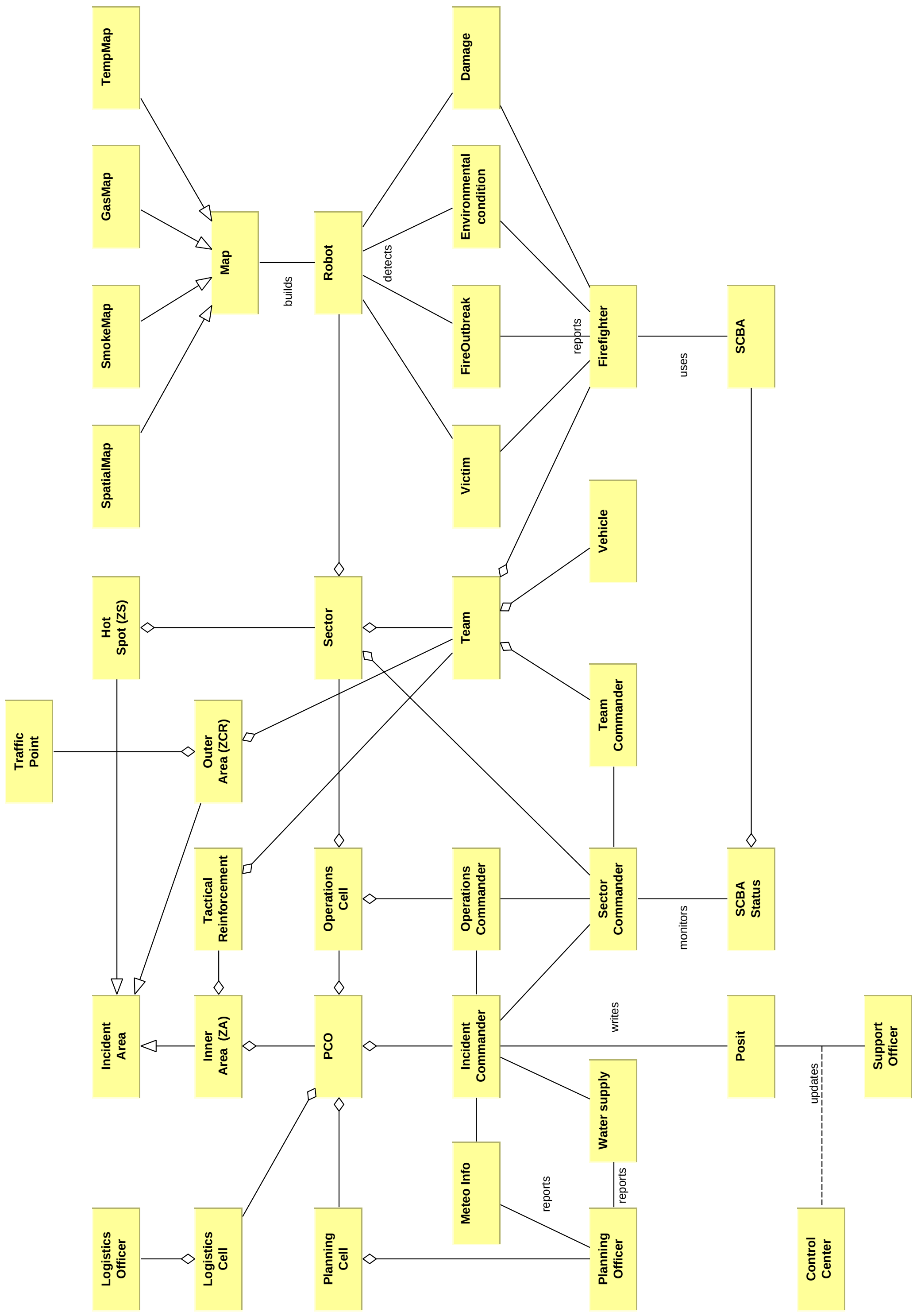


FIGURE A.8: Domain class Diagram.

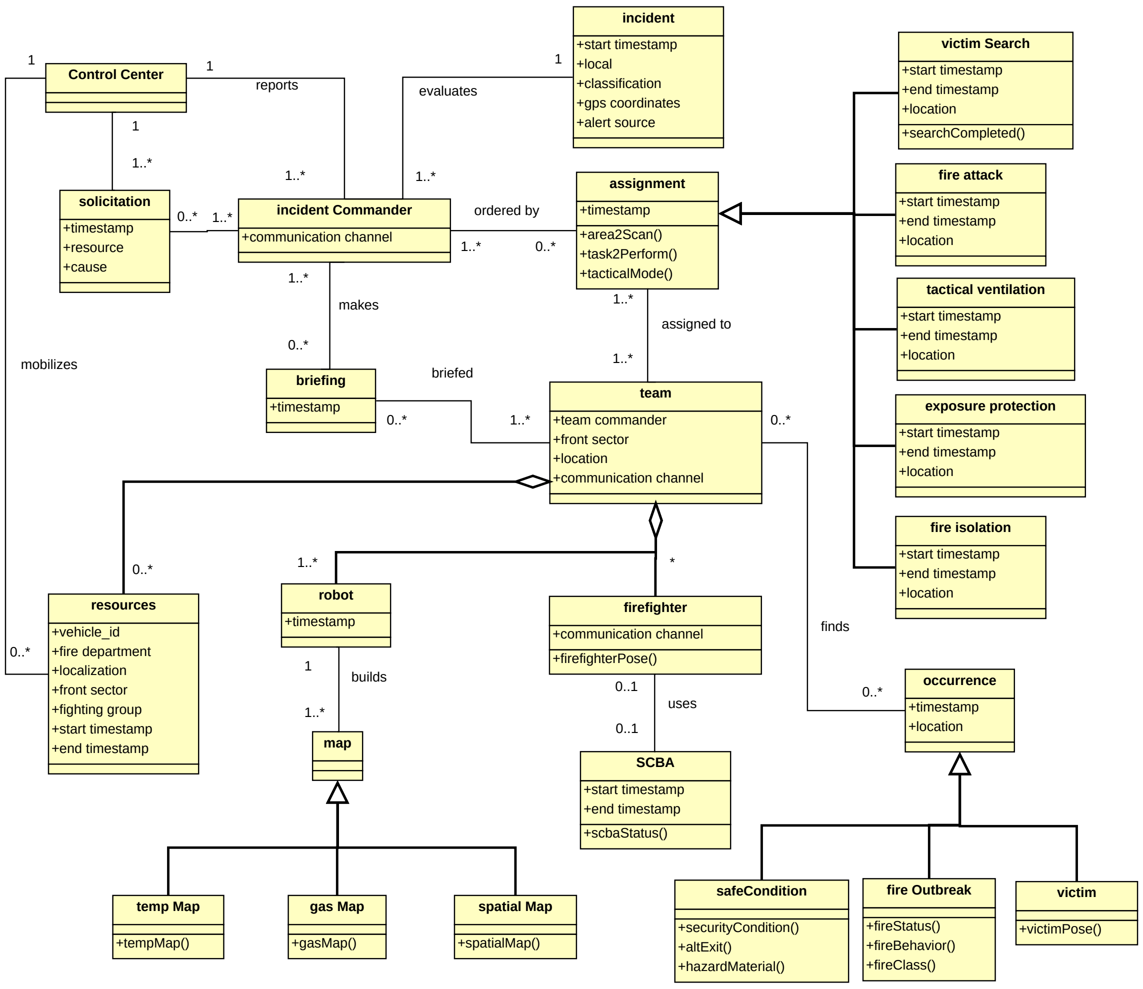


Figure A.9: Fire Attack and Search and Rescue class diagram.

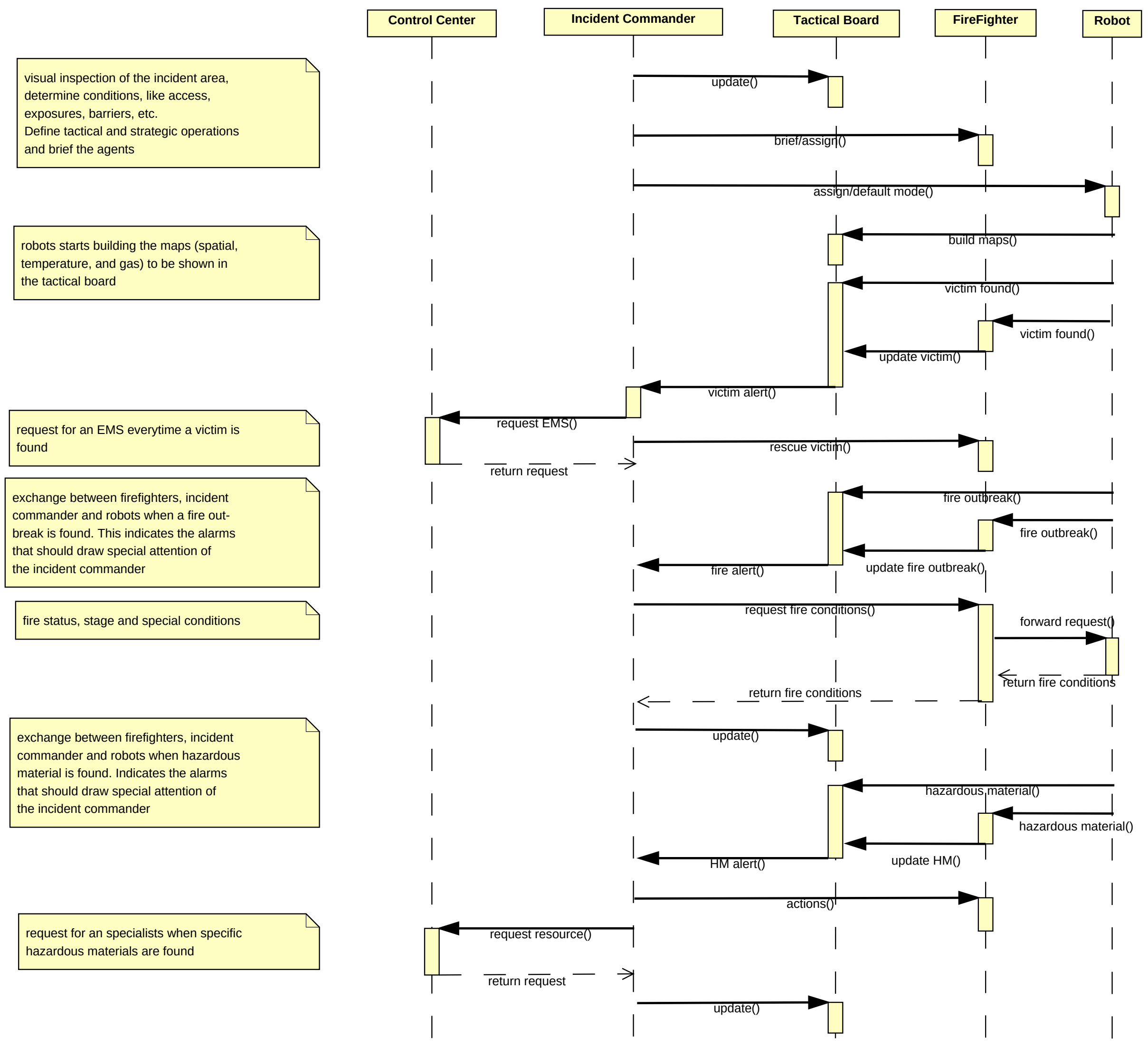

Control Center
Incident Commander
Tactical Board
FireFighter
Robot
visual inspection of the incident area, determine conditions, like access, exposures, barriers, etc. Define tactical and strategic operations and brief the agents
update()
brief/assign()
assign/default mode()
robots starts building the maps (spatial, temperature, and gas) to be shown in the tactical board
build maps()
victim found()
victim found()
update victim()
victim alert()
request for an EMS everytime a victim is found
request EMS()
return request
rescue victim()
exchange between firefighters, incident commander and robots when a fire out-break is found. This indicates the alarms that should draw special attention of the incident commander
fire outbreak()
fire outbreak()
fire alert()
update fire outbreak()
fire status, stage and special conditions
request fire conditions()
forward request()
return fire conditions
return fire conditions
update()
exchange between firefighters, incident commander and robots when hazardous material is found. Indicates the alarms that should draw special attention of the incident commander
hazardous material()
hazardous material()
HM alert()
update HM()
actions()
request for an specialists when specific hazardous materials are found
request resource()
return request
update()

FIGURE A.10: Sequence Diagram of the Fire Attack and Search and Rescue operations.

Appendix B

Messages Definition

B.1 Command and Control Messages

Command and Control messages class allow the Command Center (CC) to change the goals of the mission, define new areas of interest and start a new mission for a robot, from a predefined location. During a search and rescue mission, scanned areas are signalized to avoid duplicated tasks.

B.1.1 *start2Scan()*

Description

Coordinates of the starting point of the robot in its exploit mission.

CC ⇨ robot

Message to define the starting point of the robots' mission when its deployed on an entrance of the structure to exploit, submitted only when required by the CC. The coordinates of the starting point is obtained by clicking on the desired point on Google Maps, and the robot must complete the initial pose from its magnetic compass.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    geometry_msgs/Pose  pose
```

Code Excerpt B.1: *ROS message: start2scan().*

Json Message

```
{ "topic": "cc_0/start2scan",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": 1,
            "seq": 232
        },"pose": {
            "position": {
                "x": 0.0017761230701580644,
                "y": -0.0019882202614098787,
                "z": 0.0
            },"orientation": {
                "y": ,
                "x": ,
                "z": ,
                "w":
            }
        }
    },"op": "publish"
}
```

Code Excerpt B.2: *Json message: start2scan().*

B.1.2 *area2Scan()*

Description

Coordinates of an area of interest that should be scanned immediately. If one robot receives two or more messages of areas to scan, it should prioritize the orders addressing to the defined level of shared interaction and task criticality.

CC ⇨ firefighter; **CC ⇨ robot**; **firefighter ⇨ robot**

Message to define a new area to scan, submitted only when required by the CC or the firefighter in the hot zone. This message is also used with the *site_view()* message to scan the area with the robot video camera.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    geometry_msgs/Pose  pose
```

Code Excerpt B.3: *ROS message: area2scan().*

Json Message

```
{ "topic": "robot_1/area2scan",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": 1,
            "seq": 232
        },"pose": {
            "position": {
                "x": 0.0017761230701580644,
                "y": -0.0019882202614098787,
                "z": 0.0
            },"orientation": {
                "y": ,
                "x": ,
                "z": ,
                "w":
            }
        }
    },"op": "publish"
}
```

Code Excerpt B.4: *Json message: area2scan().*

B.1.3 *task2Perform()*

Task to perform - Action that should be carried out by the agent on the ground, whether it is a firefighter or a robot.

CC ⇨ firefighter; **CC ⇨ robot**

Action to be performed by firefighters and/or robots. Submitted only when required.

Ros Message

Ros type	**Description**
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
nav_msgs/Path	An array of poses that represents a path for a robot/firemen to follow.
actionlib_msgs/GoalStatus	*GoalID, status, PENDING, ACTIVE, PREEMPTED, SUCCEEDED, REJECTED, ABORTED, ...*

```
    Header                       header
    std_msgs/String              agent_class
    std_msgs/Int8                id
    nav_msgs/Path                path
    actionlib_msgs/GoalStatus    goal_status
```

Code Excerpt B.5: *ROS message: task2perfom().*

Json Message

```
{   "topic": "robot_1/task2perform",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "string",
            "seq": 232
        },"path": [{
                "pose": {
                    "header": {
                      "stamp": {
                        "secs": 1358783297,
                        "nsecs": 415561250
                      },"frame_id": "string",
                        "seq": 232
                    },"position": {
                        "x": 0.0017761230701580644,
                        "y": -0.0019882202614098787,
                        "z": 0.0
                    },"orientation": {
                        "y": 0.0,
                        "x": 0.0,
                        "z": -0.0001439173933738437,
                        "w": 0.9999999896438919
                    }
                }
            }
        ],"goal_status": {
            "id": "string",
            "status": 5,
            "text": "string"
        }
    },"op": "publish"
}
```

Code Excerpt B.6: *Json message: task2perform().*

B.1.4 *tacticalMode()*

Ground teams operation mode (e.g. ”defensive”, ”offensive” or ”transactional”) define by the CCO.

CC ⇨ firefighter; CC ⇨ robot

Message submitted only when required, to change the teams operating mode due to tactics change.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
actionlib_msgs/GoalID	The stamp should store the time at which this goal was requested.

```
    Header                  header
    std_msgs/String         agent_class
    std_msgs/Int8           id
    actionlib_msgs/GoalID   goal_id
```

Code Excerpt B.7: *ROS message: tacticalMode().*

Json Message

```
{
    "topic": "cc_0/tactical_mode",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "goal_id": {"stamp" : "time",
                    "id"    : "string"
                   }
    },
    "op": "publish"
}
```

Code Excerpt B.8: *Json message: tacticalMode().*

B.1.5 *searchCompleted()*

Notifies a complete area search, with or without victims, to prevent other agents to duplicate the task.

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Message responsible to prevent duplicated searches, optimizing the use of resources. Should be sent whenever the area search is completed.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
nav_msgs/GridCells	An array of cells in a 2D grid.
actionlib_msgs/GoalStatus	*GoalID, status, PENDING , ACTIVE, PREEMPTED, SUCCEEDED, REJECTED, ABORTED, ...*

```
    Header                        header
    std_msgs/String               agent_class
    std_msgs/Int8                 id
    nav_msgs/GridCells            searched_area
    actionlib_msgs/GoalStatus     goal_status
```

Code Excerpt B.9: *ROS message: searchCompleted().*

Json Message

```
{
    "topic": "robot_1/searchCompleted",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "searched_area": {
            "header": {
                "seq"  : uint32,
                "stamp": time,
                "frame_id": string
                },
            "cell_witdh" : float32,
            "cell_height": float32,
            "cells" : [{"x":float64,
                        "y":float64,
                        "z":float64
                      }]
            },
        "goal_status": {
            "id": "string",
            "status": 5,
            "text": "string"
        }
    },
    "op": "publish"
}
```

Code Excerpt B.10: *Json message: searchCompleted().*

B.2 Location messages

Information class messages are divided into two classes: conditions messages - to help classify the incident and define tactical strategies of operation; and location messages - to locate the agents, victims and relevant events in the incident zone.

B.2.1 *robotPose()*

Pose coordinates of the robot on the ground.

robot ⇨ firefighter; **robot ⇨ CC**

Message to send pose coordinates of the robot. This is a priority message and should be sent frequently, to provide CC and fire fighters with situation awareness of the robot.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    geometry_msgs/Pose  pose
```

Code Excerpt B.11: *ROS message: robotPose().*

Json Message

```
{
    "topic": "robot_1/robot_pose",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": 1,
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        }
    },
    "op": "publish"
}
```

Code Excerpt B.12: *Json message: robotPose().*

B.2.2 *firefighterPose()*

Firefighter pose - pose coordinates of the firefighter on the ground and its way of progression (e.g. standing, climbing, descending, crouched, ”on all fours”).

firefighter ⇨ robot; **firefighter ⇨ CC**

Message to send pose coordinates of the firefighter. This is a priority message and should be sent frequently, to provide the CC and robots with situation awareness of the firefighter.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinate frame and time stamp.
std_msgs/Int16.	Integers type message, to enumerate fireman state, id, etc.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8      id
    geometry_msgs/Pose pose
    float32            situation
```

Code Excerpt B.13: *ROS message: firefighterPose().*

Json Message

```
{
    "topic": "firefighter_1/firefighterPose",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        },
        "situation": "number"
    },
    "op": "publish"
}
```

Code Excerpt B.14: *Json message: firefighterPose().*

B.3 Status messages

B.3.1 *batteryStatus()*

Robot battery level. Message defined to monitor the robot battery level, to allow the CC to provide a safe exit for the robot.

robot ⇨ CC

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
pr2_msgs/PowerState	This message communicates the state of the robot's power system.

```
    std_msgs/Header      header
    std_msgs/String      agent_class
    std_msgs/Int8        id
    pr2_msgs/PowerState  status
```

Code Excerpt B.15: *ROS message: batteryStatus().*

Json Message

```
{
    "topic": "robot_1/batteryStatus",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "status": {
            "power_consumption": "number",
            "time_remaining": "number",
            "prediction_method": "string",
            "relative_capacity": "number"
        }
    }
    "op": "publish"
}
```

Code Excerpt B.16: *Json message: batteryStatus().*

B.3.2 *scbaStatus()*

Firefighters SCBA level and mission time using SCBA.

firefighter ⇨ CC

Message defined to monitor the firefighters SCBA level and mission time, to allow the CC to provide a safe exit for the firefighter.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
pr2_msgs/PowerState	This message communicates the state of the scba system.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    pr2_msgs/PowerState status
```

Code Excerpt B.17: *ROS message: scbaStatus().*

Json Message

```
{
    "topic": "firefighter_1/scbaStatus",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "status": {
            "power_consumption": "number",
            "time_remaining": "number",
            "action": "string",
            "relative_capacity": "number"
        }
    }
    "op": "publish"
}
```

Code Excerpt B.18: *Json message: scbaStatus().*

B.3.3 *teamComp()*

Description

Coordinates of an area of interest that should be scanned immediately.

CC ⇨ robot

Message to define the starting point of the robots' mission when its deployed on an entrance of the structure to exploit, submitted only when required by the CC. The coordinates of the starting point is obtained by clicking on the desired point on Google Maps, and the robot must complete the initial pose from its magnetic compass.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
std_msgs/String[]	An array of string with the identifiers of the team-mates of the agent.

```
    Header               header
    std_msgs/String      agent_class
    std_msgs/Int8        id
    std_msgs/String[]    team
```

Code Excerpt B.19: *ROS message: teamComp().*

Json Message

```
{
    "topic": "robot_1/teamComp",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "team": ["robot_2",
                 "robot_5",
                 "firefighter_3"
                ]
    },
    "op": "publish"
}
```

Code Excerpt B.20: *Json message: teamComp().*

B.3.4 *sharedLevel()*

Description

Coordinates of an area of interest that should be scanned immediately.

CC ⇨ robot

Message to define the starting point of the robots' mission when its deployed on an entrance of the structure to exploit, submitted only when required by the CC. The coordinates of the starting point is obtained by clicking on the desired point on Google Maps, and the robot must complete the initial pose from its magnetic compass.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
std_msgs/Int8	An integer to indicate the shared level of interaction with the operators. *e.g. 1- 1H:1R; 2- 1H:RT; 3- 1H:MR; 4- HT:1R; . . .*, where 1H: one human, 1R: one robot, MR: multiple robots, RT: robot team and HT: human team.

```
    Header              header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    std_msgs/Int8       sharedlevel
```

Code Excerpt B.21: *ROS message: sharedLevel()*

Json Message

```
{
    "topic": "robot_1/gas_map",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "string",
            "seq": 232
        },
        "sharedLevel": number
    },
    "op": "publish"
}
```

Code Excerpt B.22: *Json message: sharedLevel().*

B.3.5 ***netStats()***

Network Stats - message with network connection metrics from the node to the CCO and from the node to their neighbours.

robot ⇨ CC; **robot ⇨ firefighter**

This message should be sent whenever there is a network connection.

Ros Message

Ros type	**Description**
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.

ROS message type to be defined.

```
    Header                          header
    std_msgs/String                 agent_class
    std_msgs/Int8                   id
    geometry_msgs/Vector3Stamped    vector
    nav_msgs/Path                   poses
    actionlib_msgs/GoalStatus       goal_status
```

Code Excerpt B.23: *ROS message: netStats().*

Json Message

```
{
    "topic": "network/stats",
    "msg": {
        "agent_class" : "cco",
        "id" : 0,
        "quality": {
            "link": "number" ,
            "level": "number",
            "noise": "number"
        },
        "metrics": {
            "tx": "number",
            "rx": "number"
        },
        "neighbors:{[
            "items": {
                "ip": "string",
                "quality":{
                    "link": "number",
                    "level": "number",
                    "noise": "number"
                },
                "metrics": {
                    "tx": "number",
                    "rx": "number"
                }
            }
        ]}
    },
    "op": "publish"
}
```

Code Excerpt B.24: *Json message: netStats().*

B.3.6 *envStatus()*

Message to classify the environment status. Aggregates the messages FireClass, FireStage, FireBehaviour to reduce the number of published

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Information message to indicate combustible materials feeding the fire. This message should be sent whenever a fire outbreak is found.

Ros Message

geometry_msgs/PoseStamped - A Pose with reference coordinates frame and time stamp.
std_msgs/Int16MultiArray - Simple array to fire class designation.

```
Header header
geometry_msgs/PoseStamped pose
std_msgs/Int16MultiArray hazard_conditions
std_msgs/Int16MultiArray temp_conditions
std_msgs/Int16MultiArray gas_condition
```

Code Excerpt B.25: *ROS message: envStatus().*

Json Message

```
{
    "topic": "robot_1/env_status",
    "msg": {
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },"frame_id": "map",
            "seq": 232
        },pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },"orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        },"fire_class":[{"items": 3}],
          "hazard_conditions": [{"items": 2}],
          "temp": [{"items": 2}],
          "gas": [{"items": 4}]
    },"op": "publish"
}
```

Code Excerpt B.26: *Json message: envStatus().*

B.3.7 *fireStatus()*

Fire class - classification of fire due to combustible materials.

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Information message to indicate combustible materials feeding the fire. This message should be sent whenever a fire outbreak is found.

Ros Message

geometry_msgs/PoseStamped - A Pose with reference coordinates frame and time stamp.
std_msgs/Int16MultiArray - Simple array to fire class designation.

This message is deprecated. Included in the message envStatus

B.3.8 *fireStage()*

Indicates the state of the fire at a given location (e.g. *"incipient"*, *"smoldering"*, *"growth"*, *"fully developed"*, *"decay"*).

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Information message to indicate the stage of a fire outbreak, determining the evolution of the incident and safety conditions. This message should be sent whenever a fire outbreak is found.

Ros Message

geometry_msgs/PoseStamped - A Pose with reference coordinate frame and time stamp.
std_msgs/Int16 - Simple float type message, to enumerate situation.

Json Message

```
{
    "topic": "robot_1/fire_stage",
    "msg": {
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        } ,
        "fire_stage": 2
    },
    "op": "publish"
}
```

Code Excerpt B.27: *Json message: fireStage().*

B.3.9 *fireBehaviour()*

Indicates the fire behavior based on predetermined conditions and in a given location (e.g. *"backdraft", "rollover", "flashover"*).

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Complementary information messages to predict fire behavior, determining safety conditions. This message should be sent whenever one of these conditions is found.

Ros Message

geometry_msgs/PoseStamped - A Pose with reference coordinates frame and time stamp.

Json Message

```
{
    "topic": "robot_1/fire_behaviour",
    "msg": {
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        } ,
        "fire_behaviour": 2
    },
    "op": "publish"
}
```

Code Excerpt B.28: *Json message: fireBehaviour().*

B.4 Conditions messages

B.4.1 *securityCondition()*

Security conditions - Indicates the firefighter if it is safe to move in a direction or enter an area for the known fire conditions. (E.g. *"climb above a fire without a charged hose - unsafe, do not proceed"*).

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Message to define safety conditions of an area which is about to be scanned by a firefighter. This message should be sent while firefighters move forward on the ground.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinate frame and time stamp.
std_msgs/Int16	Simple float type message, to enumerate situation.

```
    Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
geometry_msgs/PoseStamped[] poses
```

Code Excerpt B.29: *ROS message: securityConditions().*

Json Message

```
{
    "topic": "robot_1/securityConditions",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        } ,
        "security_conditions": 2
    },
    "op": "publish"
}
```

Code Excerpt B.30: *Json message: securityConditions().*

B.4.2 *altExit()*

Detection of doors and/or windows that allow the escape of fire in adverse situations. Always depends on the maps defined by agents nearby.

robot ⇨ firefighter; **robot ⇨ CC**; **firefighter ⇨ CC**

Information message to signal a room secondary exit or an egress exit determining a safety path for firefighters and victims. This message should be sent whenever one of these exits is found.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.

Continued on next page

Table B.14 – *Continued from previous page*

Ros type	Description
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.
std_msgs/Int16	Simple float type message, to enumerate situation.

```
Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
geometry_msgs/Pose pose
float32 gas
```

Code Excerpt B.31: *ROS message: altExit.*

Json Message

```
{
    "topic": "robot_1/altExit",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "pose": {
            "position": {
                "y": -0.0019882202614098787,
                "x": 0.0017761230701580644,
                "z": 0.0
            },
            "orientation": {
                "y": 0.0,
                "x": 0.0,
                "z": -0.0001439173933738437,
                "w": 0.9999999896438919
            }
        },
        poses: [
            {"pose": {
                "header": {
                    "stamp": {
                        "secs": 1358783297,
                        "nsecs": 415561250
                    },
                    "frame_id": "string",
                    "seq": 345
                },
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            },
            "exit": 0
        }]
    },
    "op": "publish"
}
```

Code Excerpt B.32: *Json message: altExit().*

B.4.3 *siteView()*

Image sequence to allow the human user the fire conditions and/or identify a victim condition.

robot ⇨ firefighter; **robot ⇨ CC**

Image message to provide primary source of surroundings awareness to the CC and firefighters, for victim search and safety conditions.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
sensor_msgs/CompressedImage	This message contains a compressed image.

```
    Header                      header
    std_msgs/String             agent_class
    std_msgs/Int8               id
    sensor_msgs/CompressedImage image
```

Code Excerpt B.33: *ROS message: siteView().*

Json Message

```
{
    "topic": "robot_1/siteView",
    "msg": {
        "agent_class" : "cco",
        "id" : 0,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "string",
            "seq": 232
        },
        "image":{
            "header": {
                "stamp": {
                    "secs": 1358783297,
                    "nsecs": 415561250
                },
                "frame_id": "string",
                "seq": 232
            }
            "format": "string",
            "data":["number"]
        }
    },
    "op": "publish"
}
```

Code Excerpt B.34: *Json message: siteView().*

B.5 Mapping messages

Mapping messages class are used only by robots providing information about the interior of the building, aiding the CC to establish eventual hazardous conditions for the firefighters and victims.

B.5.1 *compressedMap()*

Spatial map - 2D grid map to identify the location of the agents in the building, entry and egress accesses and respective conditions.

robot ⇨ CC; **robot ⇨ fireghter**; **CC ⇨ firefighter**

Priority message to build the interior map and provide the CC with situation awareness of the incident. This message should be sent whenever there is a network connection.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
nav_msgs/MapMetaData	This represents the map *metadata* information.
geometry_msgs/Point	A point to represent top-left coordinate of the transmitted window
geometry_msgs/Point	A point to represent bottom-right coordinate of the transmitted window
std_msgs/ByteMultiArray	A byte array with the occupancy-grid compressed data.

```
Header header
std_msgs/String    agent_class
std_msgs/Int8      id
nav_msgs/MapMetaData info
geometry_msgs/Point iniwindow
geometry_msgs/Point endwindow
std_msgs/ByteMultiArray data
```

Code Excerpt B.35: *ROS message: compressedMap().*

Json Message

```
{
    "topic": "robot_1/compressedMap",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "iniwindow": {  "x": number,
                        "y": number,
                        "z": 0 },
        "endwindow": {  "x": number,
                        "y": number,
                        "z": 0 },
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.36: *Json message: compressedMap().*

B.5.2 *tempMap()*

Temperature map - 2D grid map with temperature levels to identify hazardous conditions in the area.

robot ⇨ CC; **robot ⇨ firefighter**; **CC ⇨ firefighter**

Priority message to build the temperature map and provide the CC with heat and hazardous awareness. This message should be sent whenever there is a network connection.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
nav_msgs/OccupancyGrid	This represents a 2-D grid map, in which each cell represents the probability of occupancy.

```
Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
geometry_msgs/Pose pose
float32 temp
```

Code Excerpt B.37: *ROS message: tempMap().*

This message is deprecated. Included in the message envStatus

Json Message

```
{
    "topic": "robot_1/tempMap",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.001988220261409878 7,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.38: *Json message: tempMap().*

B.5.3 *gasMap()*

Gas map - 2D grid map with the type of gas and respective concentration (e.g. CO, CO_2[organic material], HCN [acrylic fibers, polyurethane or nylon], HCI and $COCI_2$ [PVC, vinyl wallpaper and cable installation].

robot ⇨ CC; **robot ⇨ firefighter**; **CC ⇨ firefighter**

Priority message to build the temperature map and provide the CC with gas concentration and hazardous awareness. This message should be sent whenever there is a network connection.

Ros Message

Ros type	**Description**
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.

Continued on next page

Table B.18 – *Continued from previous page*

Ros type	**Description**
nav_msgs/OccupancyGrid	This represents a 2-D grid map, in which each cell represents the probability of occupancy.

```
Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
geometry_msgs/Pose pose
float32 gas
```

Code Excerpt B.39: *ROS message: gasMap().*

This message is deprecated. Included in the message envStatus

Json Message

```
{
    "topic": "robot_1/gasMap",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.40: *Json message: gasMap().*

B.5.4 *victimFound()*

Victim pose - pose coordinates of the victim on the ground and its state condition (e.g. *"aware", "semi-aware", "unconscious", "unknown"*), situation condition (e.g. *"surface", "trapped", "entombed", "unknown"*).

robot ⇨ CC; **robot ⇨ firefighter**; **firefighter ⇨ CC**

Message to send pose coordinates of the victim. This is a priority message to be sent whenever a victim is found, and provide the CC and firefighters with situation awareness of the victim.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.
std_msgs/Int16MultiArray	Array of integers type message, to enumerate victim state,id,etc.

```
Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
    geometry_msgs/PoseStamped[] poses
    std_msgs/Int16MultiArray status
```

Code Excerpt B.41: *ROS message: victimFound().*

Json Message

```
{
    "topic": "robot_1/victimFound",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "victims": [{
                "header": {
                    "stamp": {
                        "secs": 1358783297,
                        "nsecs": 415561250
                    },
                    "frame_id": "map",
                    "seq": 232
                },
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }]
            } ,
        "status":[{
            "state"    : number,
            "situation": number,
            "condition": number,
            "type"     : number
        }]
    },
    "op": "publish"
}
```

Code Excerpt B.42: *Json message: victimFound().*

B.5.5 *smokeMap()*

Hazard condition - position of materials and hazardous conditions (e.g. containers of flammable or explosive materials near fire outbreak that requires special attention or imminent "*BLEVE*").

robot ⇨ CC; **robot ⇨ firefighter**; **firefighter ⇨ CC**

Message to send location coordinates of the hazardous materials and conditions. This is a priority message and should be sent whenever these conditions are found, to provide the CC and firefighters with situation awareness of the overall safety conditions.

Ros Message

Ros type	**Description**
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	Simple float type message, to enumerate situation.
std_msgs/Int16	*This message is deprecated. Included in the message envStatus*

A Pose with reference coordinate frame and time stamp.

```
    Header header
    std_msgs/String     agent_class
    std_msgs/Int8       id
    geometry_msgs/PoseStamped[] poses
```

Code Excerpt B.43: *ROS message: smokeMap().*

Json Message

```
{
    "topic": "robot_1/smokeMap",
    "msg": {
        "agent_class" : "cco",
        "id" : 0,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.001988220261409878 7,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.44: *Json message: smokeMap().*

B.5.6 *hazardMat()*

Hazard condition - position of materials and hazardous conditions (e.g. containers of flammable or explosive materials near fire outbreak that requires special attention or imminent " *BLEVE*").

robot ⇨ CC; **robot ⇨ firefighter**; **firefighter ⇨ CC**

Message to send location coordinates of the hazardous materials and conditions. This is a priority message and should be sent whenever these conditions are found, to provide the CC and firefighters with situation awareness of the overall safety conditions.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinate frame and time stamp.
std_msgs/Int16	Simple float type message, to enumerate situation.

This message is deprecated. Included in the message envStatus

```
    Header header
    std_msgs/String    agent_class
    std_msgs/Int8      id
    geometry_msgs/Pose pose
    float32 temp
```

Code Excerpt B.45: *ROS message: hazardMat().*

Json Message

```
{
    "topic": "robot_1/hazardMap",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.46: *Json message: hazardMat().*

B.5.7 *fireOutbreak()*

Fire outbreak - coordinates of the fire outbreak, which requires immediate attention of the CC and the firefighters.

robot ⇨ CC; **robot ⇨ firefighter**; **firefighter ⇨ CC**

Message to send location coordinates of a fire outbreak. This is a priority message and should be sent whenever a new fire outbreak is found, to allow the CC and firefighters to take adequate measures.

Ros Message

Ros type	Description
std_msgs/String	A string with the class of the agent (team, robot or firefighter).
std_msgs/Int8	An integer with the number id of the agent.
geometry_msgs/PoseStamped	A Pose with reference coordinates frame and time stamp.

This message is deprecated. Included in the message envStatus

```
Header             header
std_msgs/String    agent_class
std_msgs/Int8      id
geometry_msgs/Pose pose
```

Code Excerpt B.47: *ROS message: fireOutbreak().*

Json Message

```
{
    "topic": "robot_1/fireOutbreak",
    "msg": {
        "agent_class" : "robot",
        "id" : 1,
        "header": {
            "stamp": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "frame_id": "map",
            "seq": 232
        },
        "info": {
            "map_load_time": {
                "secs": 1358783297,
                "nsecs": 415561250
            },
            "resolution": float64,
            "width": uint32,
            "height": uint32,
            "origin":{
                "position": {
                    "y": -0.0019882202614098787,
                    "x": 0.0017761230701580644,
                    "z": 0.0
                },
                "orientation": {
                    "y": 0.0,
                    "x": 0.0,
                    "z": -0.0001439173933738437,
                    "w": 0.9999999896438919
                }
            }
        } ,
        "data": [uint8]
    },
    "op": "publish"
}
```

Code Excerpt B.48: *Json message: fireOutbreak().*

B.6 Messages and HRI taxonomy

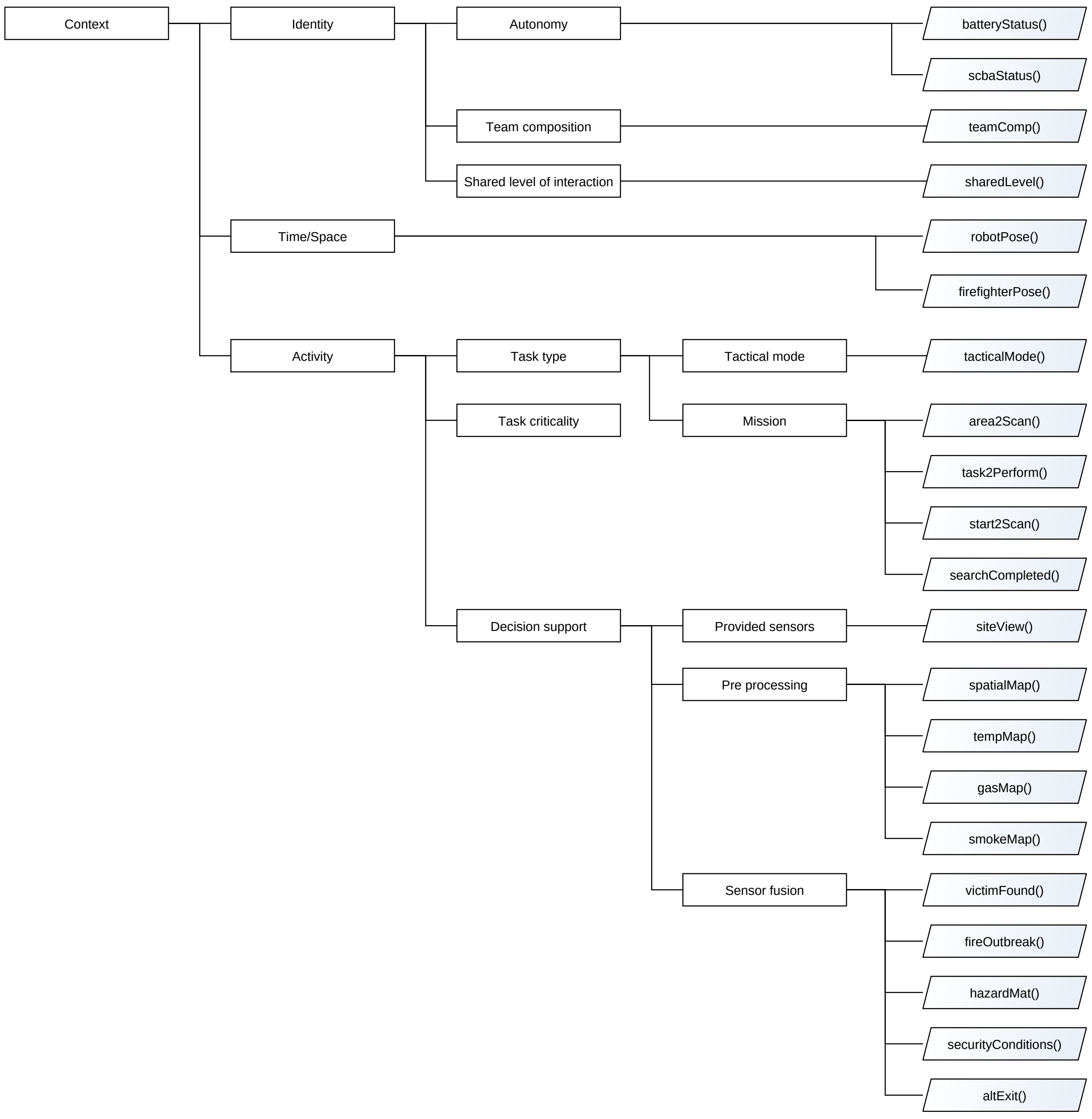


Figure B.1: Messages affecting the HRI taxonomy.

Appendix C

Entity-Relationship Diagram

C.1 Conceptual models

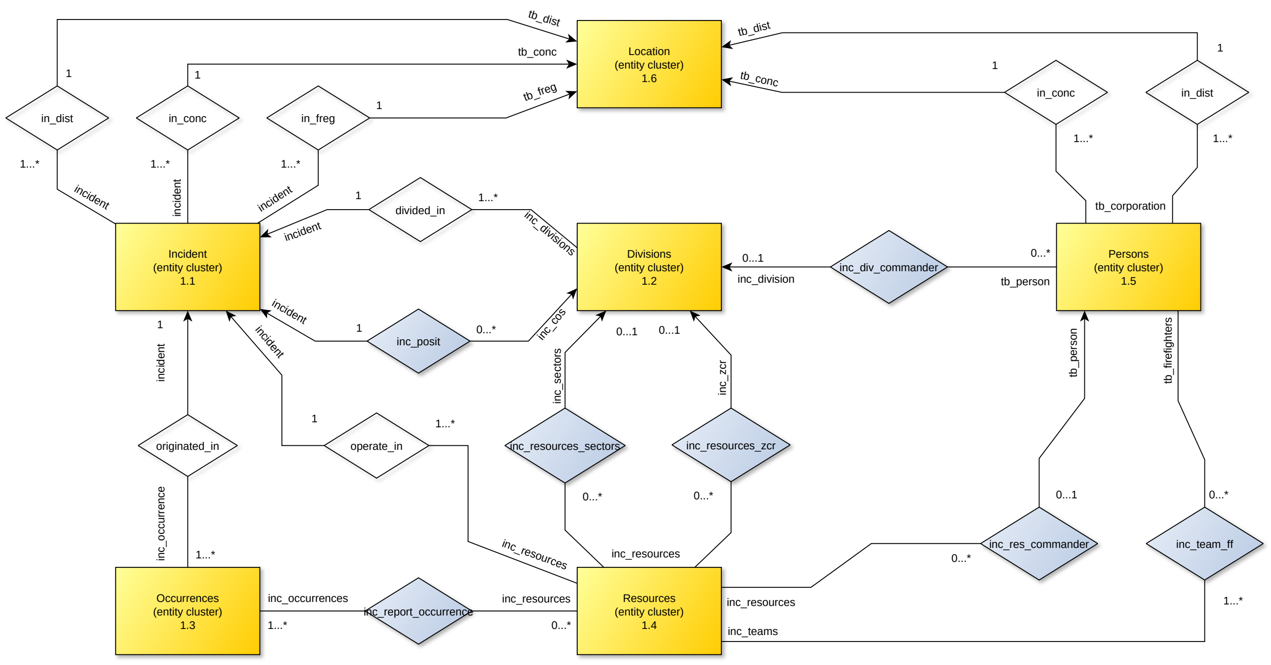


FIGURE C.1: Database conceptual schema - *root entity cluster*.

C.1.1 *Incident cluster*

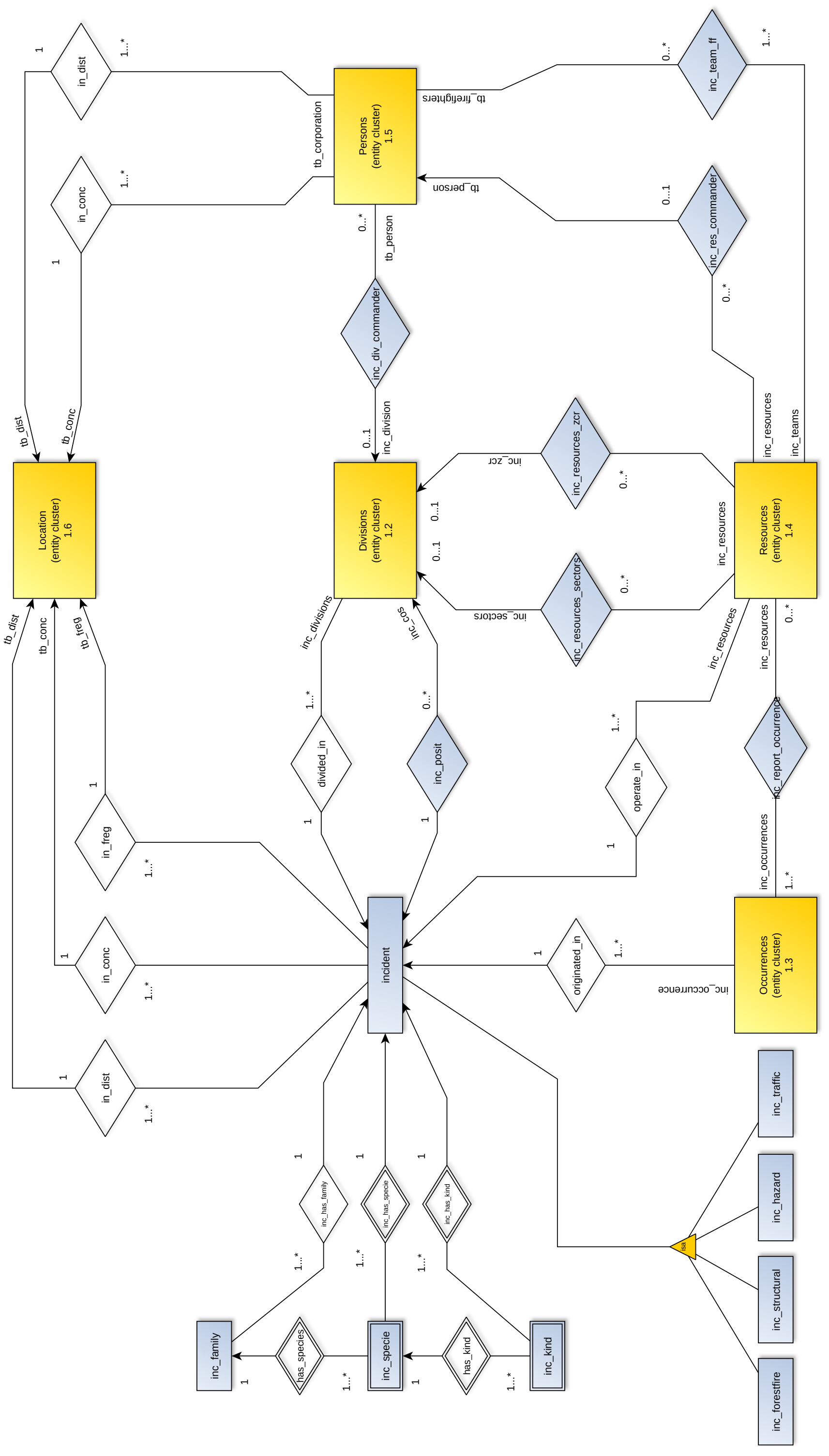


FIGURE C.2: Cluster 1.1 - Incident conceptual model.

C.1.2 *Incident divisions cluster*

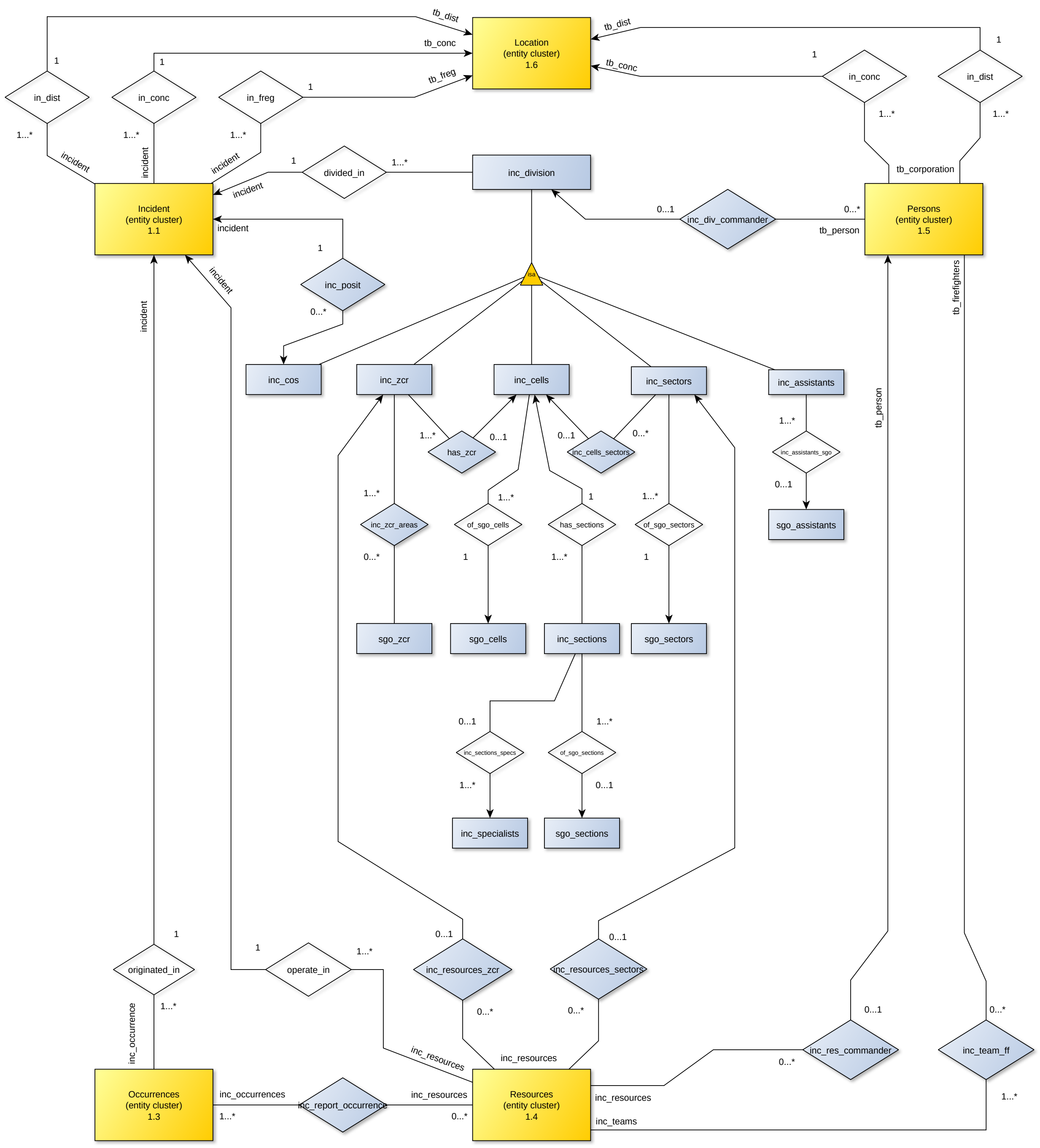


FIGURE C.3: Cluster 1.2 - Incident divisions conceptual model.

C.1.3 *Incident occurrences cluster*

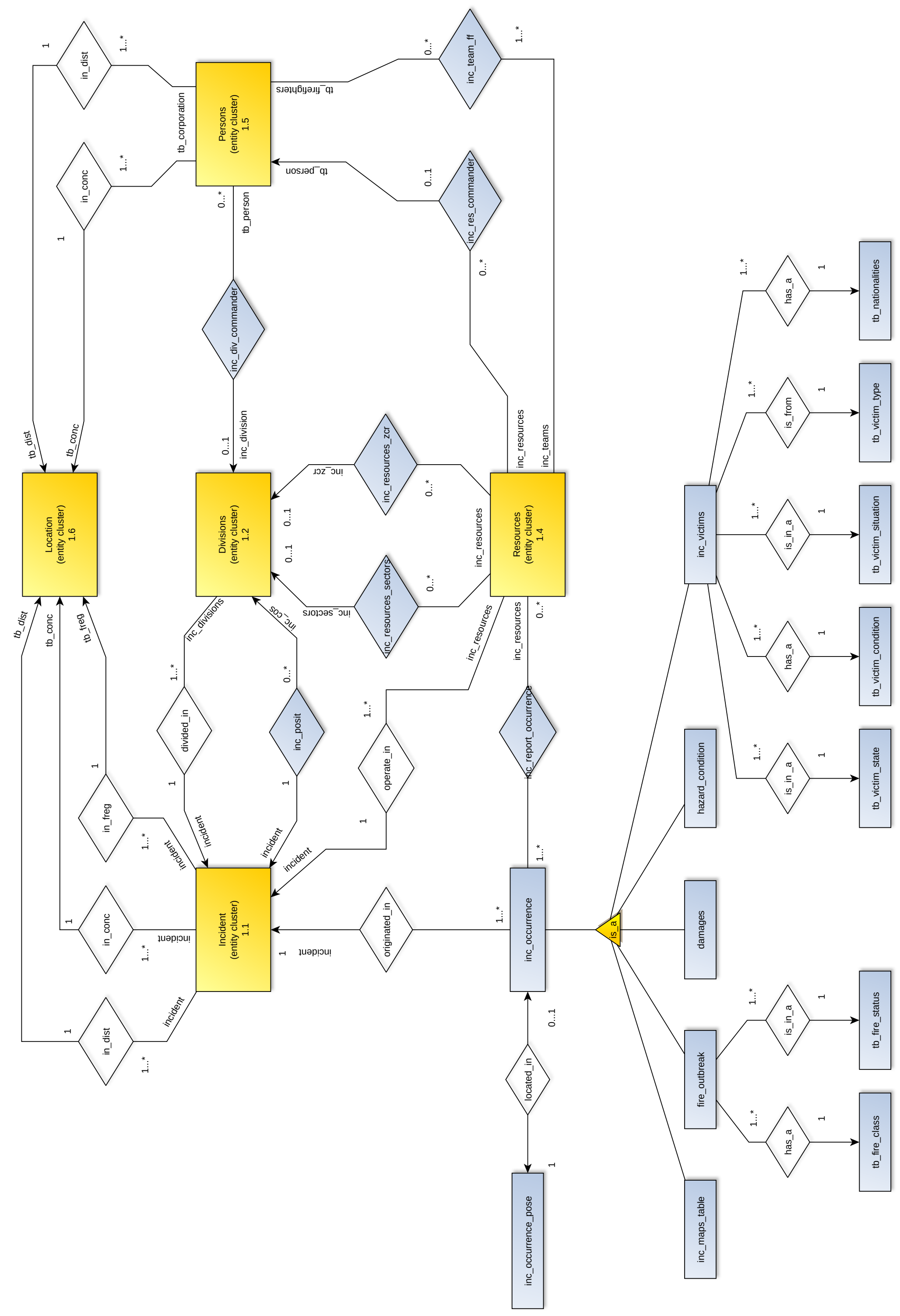


FIGURE C.4: Cluster 1.3 - Incident occurrences conceptual model.

C.1.4 *Incident resources cluster*

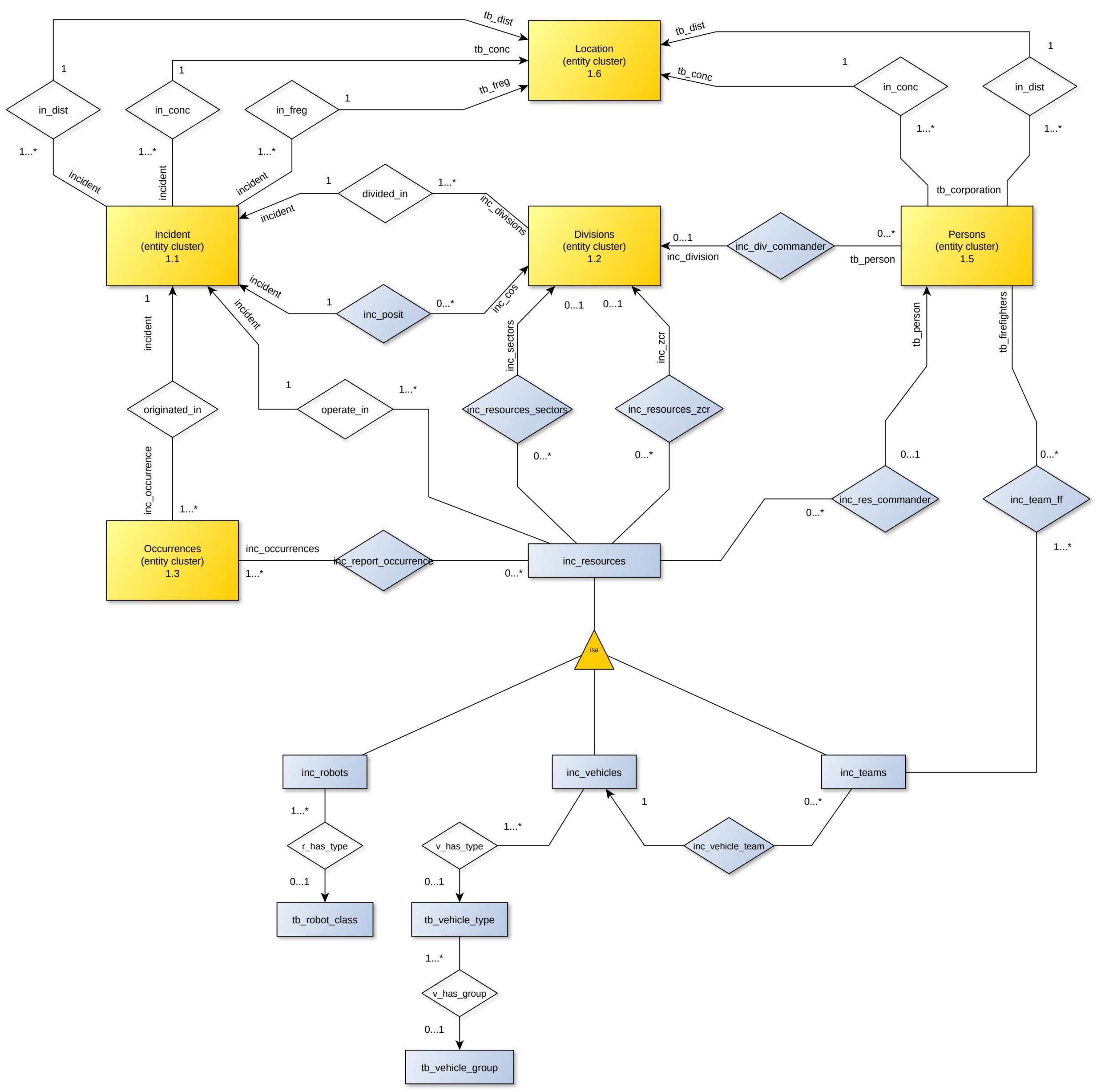


FIGURE C.5: Cluster 1.4 - Incident resources conceptual model.

C.1.5 *Incident personnel cluster*

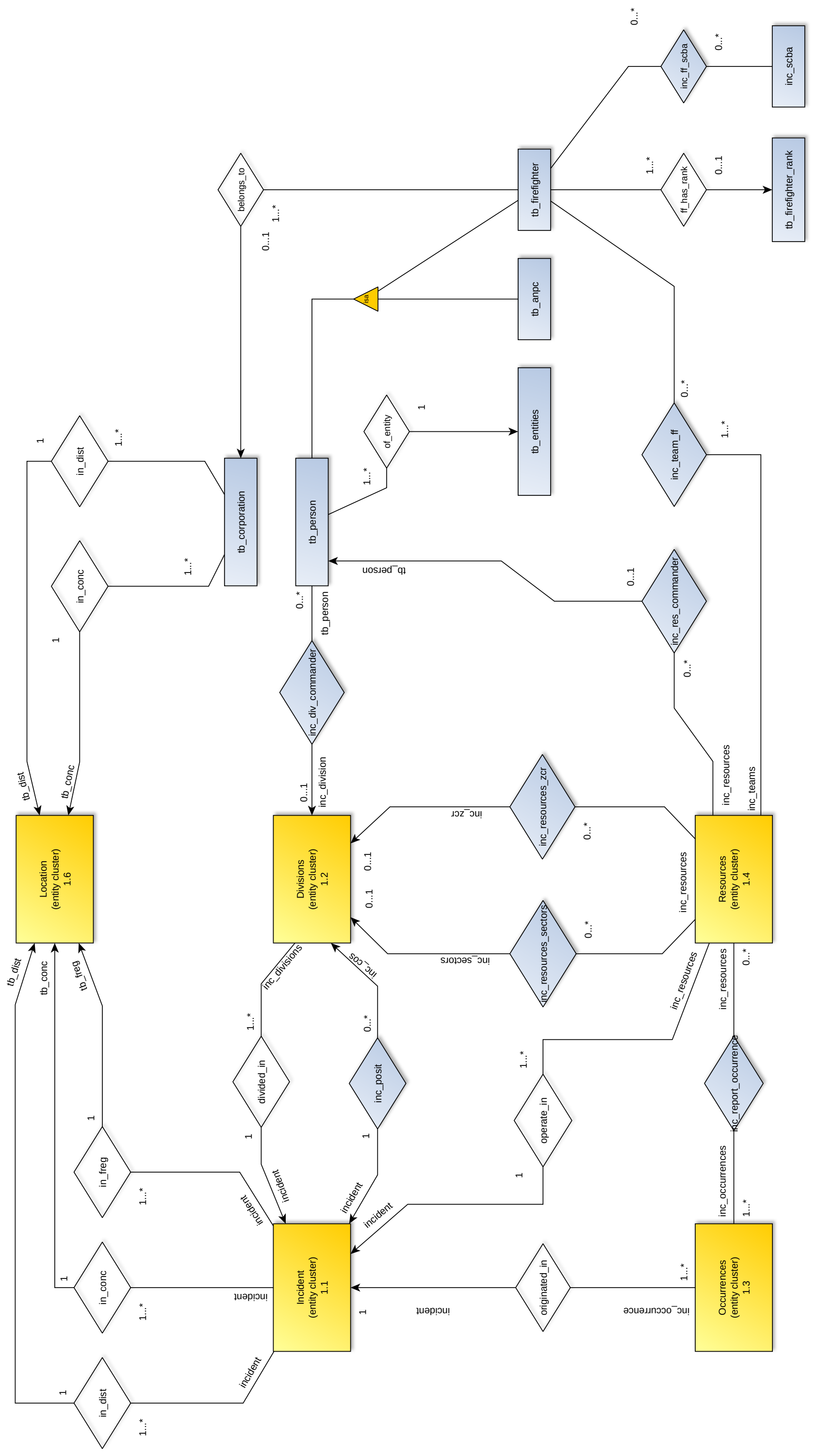


FIGURE C.6: Cluster 1.5 - Incident personnel conceptual model..

C.1.6 *Incident locations cluster*

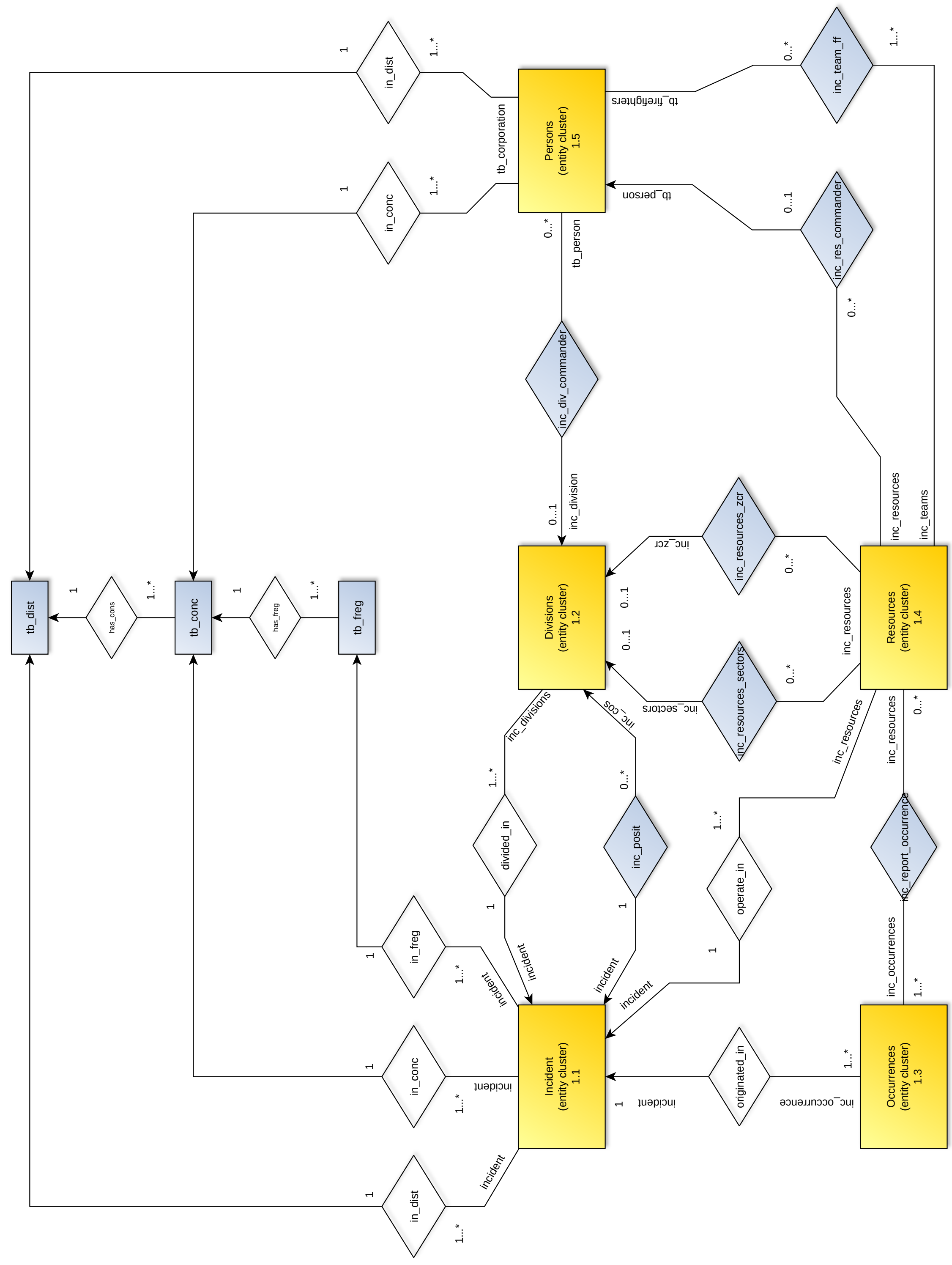


FIGURE C.7: Cluster 1.6 - Incident locations conceptual mode.

C.2 Data models

C.2.1 Incident cluster

```
CREATE TABLE `incident` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `setting_time` datetime NOT NULL,
  `local_ocorr` varchar(50) DEFAULT NULL,
  `lat_dms` varchar(20) DEFAULT NULL,
  `lng_dms` varchar(20) DEFAULT NULL,
  `id_incident_type` varchar(4) DEFAULT NULL,
  `no_ocorr` int(11) DEFAULT NULL,
  `man_channel` int(11) DEFAULT NULL,
  `info` mediumtext,
  `id_tb_freg` int(11) DEFAULT NULL,
  `id_tb_conc` int(11) DEFAULT NULL,
  `id_tb_dist` int(11) DEFAULT NULL,
  `id_family` int(11) DEFAULT NULL,
  `id_kind` int(11) DEFAULT NULL,
  `id_type` int(11) DEFAULT NULL,
  `id_sgo_phase` int(11) DEFAULT NULL,
  `lat_wgs84` decimal(10,7) DEFAULT NULL,
  `lng_wgs84` decimal(10,7) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `id_tb_freg` (`id_tb_freg`),
  KEY `id_tb_conc` (`id_tb_conc`),
  KEY `id_tb_dist` (`id_tb_dist`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1 COMMENT='Tabela de sinistros'

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`incident_BINS`
BEFORE INSERT ON `cco`.`incident`
FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
begin
        SET NEW.setting_time = NOW();
end
$$

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`incident_AINS`
AFTER INSERT ON `cco`.`incident`
FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
begin
        insert into timeline (setting_time, frame_id,seq,id_incident) VALUES (NEW.
    setting_time, "inicio",1,NEW.id);
        insert into inc_structural (id) VALUES (NEW.id);
end


CREATE
```

```
DEFINER=`root`@`localhost`
TRIGGER `cco`.`incident_AUPD`
AFTER UPDATE ON `cco`.`incident`
FOR EACH ROW
-- Edit trigger body code below this line. Do not edit lines above this one
begin
        IF (NEW.id_sgo_phase != OLD.id_sgo_phase) THEN
        CASE NEW.id_sgo_phase
        WHEN 1 THEN
                -- insert six teams in the incident structure
                SET @i = 0;
                WHILE (@i < 6) DO
                        insert into inc_teams (num_team, id_incident) VALUES (@i+1,NEW.id);
                        SET @i = @i + 1;
                END WHILE;
                -- insert the first COS
                insert into inc_cos (id_incident) VALUES (NEW.id);
        WHEN 2 THEN
                -- insert CECOP in the incident structure
                insert into inc_cells (id_sgo_cells, id_incident) VALUES (1,NEW.id);
        WHEN 3 THEN
                -- insert CEPLAN and CELOG and eventually CECOP in the incident structure
                insert into inc_cells (id_sgo_cells, id_incident) SELECT id, NEW.id FROM
    sgo_cells WHERE id NOT IN (SELECT id_sgo_cells FROM inc_cells WHERE id_incident=NEW.id);
                -- insert assistants
                insert into inc_assistants (id_sgo_assistants, id_incident) SELECT id, NEW.
    id FROM sgo_assistants WHERE id IN (1,2);
        WHEN 4 THEN
                -- insert CEPLAN and CELOG  and eventually CECOP in the incident structure
                insert into inc_cells (id_sgo_cells, id_incident) SELECT id, NEW.id FROM
    sgo_cells WHERE id NOT IN (SELECT id_sgo_cells FROM inc_cells WHERE id_incident=NEW.id);
                -- insert assistants
                insert into inc_assistants (id_sgo_assistants, id_incident) SELECT id, NEW.
    id FROM sgo_assistants WHERE id NOT IN (SELECT id_sgo_assistants FROM inc_assistants
    WHERE id_incident=NEW.id);
        END CASE;
        END IF;
end
```

Code Excerpt C.1: *incident*

```
CREATE TABLE  `cco`.`inc_structural` (
  `id` int(11) NOT NULL,
  `i_curso` tinyint(1) NOT NULL DEFAULT '0',
  `i_resolucao` tinyint(1) NOT NULL DEFAULT '0',
  `i_conclusao` tinyint(1) NOT NULL DEFAULT '0',
  `i_finalizado` tinyint(1) NOT NULL DEFAULT '0',
  `com_fogovista` tinyint(1) NOT NULL DEFAULT '0',
  `sem_fogovista` tinyint(1) NOT NULL DEFAULT '0',
  `em_habitacao` tinyint(1) NOT NULL DEFAULT '0',
  `em_comercio` tinyint(1) NOT NULL DEFAULT '0',
  `em_industria` tinyint(1) NOT NULL DEFAULT '0',
  `em_outro` tinyint(1) NOT NULL DEFAULT '0',
  `em_outro_desc` varchar(60) NOT NULL,
  `tipo_unifamiliar` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_hospital` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_edificio` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_militar` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_utilidade` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_outro` tinyint(1) NOT NULL DEFAULT '0',
  `tipo_outro_desc` varchar(60) NOT NULL,
  `p_horizontal` tinyint(1) NOT NULL DEFAULT '0',
  `p_vertical` tinyint(1) NOT NULL DEFAULT '0',
  `m_agua` tinyint(1) NOT NULL DEFAULT '0',
  `agua_outros` tinyint(1) NOT NULL DEFAULT '0',
  `agua_desc` varchar(60) NOT NULL,
  `ps_habitacoes` tinyint(1) NOT NULL DEFAULT '0',
  `ps_comercio` tinyint(1) NOT NULL DEFAULT '0',
  `ps_industria` tinyint(1) NOT NULL DEFAULT '0',
  `ps_outro` tinyint(1) NOT NULL DEFAULT '0',
  `faco_reconhecimento` tinyint(1) NOT NULL DEFAULT '0',
  `faco_salvamentos` tinyint(1) NOT NULL DEFAULT '0',
  `faco_meiosacao` tinyint(1) NOT NULL DEFAULT '0',
  `faco_protecao` tinyint(1) NOT NULL DEFAULT '0',
  `faco_ofensiva` tinyint(1) NOT NULL DEFAULT '0',
  `faco_defensiva` tinyint(1) NOT NULL DEFAULT '0',
  `faco_rescaldo` tinyint(1) NOT NULL DEFAULT '0',
  `faco_vigilancia` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_inc_structural_1` FOREIGN KEY (`id`) REFERENCES `incident` (`id`) ON DELETE
     CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Code Excerpt C.2: *inc_structural*

eg. *fire*, *accident*, *infra-structures*, ...

```
CREATE TABLE `cco`.`tb_incident_family` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Code Excerpt C.3: *tb_incident_family*

(eg. *Products*, *Transportation*, *Buildings*, ...)

```
CREATE TABLE `cco`.`tb_incident_specie` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_family` int(11) NOT NULL,
  `id_specie` int(11) NOT NULL,
  `design` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_tb_incident_specie_1_idx` (`id_family`),
  CONSTRAINT `fk_tb_incident_specie_1` FOREIGN KEY (`id_family`) REFERENCES `
    tb_incident_family` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Code Excerpt C.4: *tb_incident_specie*

(eg. *Schools*, *Hospitals*, *Military Facilities*, *Commercial Facilities*, *Hotels*, ...)

```
CREATE TABLE `tb_incident_kind` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_family` int(11) NOT NULL,
  `id_specie` int(11) NOT NULL,
  `id_type` varchar(2) NOT NULL,
  `design` varchar(100) NOT NULL,
  `code` varchar(4) NOT NULL,
  `info` text NOT NULL,
  `id_kind` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_tb_incident_kind_1_idx` (`id_specie`),
  CONSTRAINT `fk_tb_incident_kind_1` FOREIGN KEY (`id_specie`) REFERENCES `
    tb_incident_specie` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=96 DEFAULT CHARSET=latin1
```

Code Excerpt C.5: *tb_incident_kind*

C.2.2 Incident divisions cluster

```
CREATE TABLE `inc_divisions` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_division_type` int(11) NOT NULL,
  `id_incident` int(11) NOT NULL,
  `id_person` int(11) DEFAULT NULL,
  `rob_channel` varchar(5) DEFAULT NULL,
  `siresp_channel` varchar(5) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_divisions_1_idx` (`id_incident`),
  KEY `fk_inc_divisions_2_idx` (`id_person`),
  CONSTRAINT `fk_inc_divisions_1` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`)
    ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_divisions_2` FOREIGN KEY (`id_person`) REFERENCES `tb_person` (`id`) ON
     UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=198 DEFAULT CHARSET=latin1
```

Code Excerpt C.6: *inc_divisions*

```
CREATE TABLE `inc_div_commander` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_inc_division` int(11) DEFAULT NULL,
  `id_tb_person` int(11) DEFAULT NULL,
  `setting_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_divisions_idx` (`id_inc_division`),
  KEY `fk_tb_person_idx` (`id_tb_person`),
  CONSTRAINT `fk_tb_person` FOREIGN KEY (`id_tb_person`) REFERENCES `tb_person` (`id`) ON
    DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_inc_divisions` FOREIGN KEY (`id_inc_division`) REFERENCES `inc_divisions`
    (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Code Excerpt C.7: *inc_div_commander*

```
CREATE TABLE `inc_cos` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_incident` int(11) NOT NULL,
  `id_person` int(11) NOT NULL,
  `rob` varchar(5) DEFAULT NULL,
  `siresp` varchar(5) DEFAULT NULL,
  `setting_time` datetime NOT NULL,
  `lat_dms` varchar(45) DEFAULT NULL,
  `lng_dms` varchar(45) DEFAULT NULL,
  `lat_wgs84` decimal(10,7) DEFAULT NULL,
  `lng_wgs84` decimal(10,7) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_cos_1_idx` (`id_incident`),
  KEY `fk_inc_cos_2_idx` (`id_person`),
  CONSTRAINT `fk_inc_cos_1` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_cos_BINS`
BEFORE INSERT ON `cco`.`inc_cos`
FOR EACH ROW
begin
        SET NEW.setting_time = NOW();
end
```

Code Excerpt C.8: *inc_cos*

```
CREATE TABLE `inc_cells` (
  `id` int(11) NOT NULL,
  `id_sgo_cells` int(11) NOT NULL,
  `id_incident` int(11) NOT NULL,
  `setting_time` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_cells_2_idx` (`id_sgo_cells`),
  KEY `fk_inc_cells_3_idx` (`id_incident`),
  CONSTRAINT `fk_inc_cells_1` FOREIGN KEY (`id`) REFERENCES `inc_divisions` (`id`) ON DELETE
     CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_cells_2` FOREIGN KEY (`id_sgo_cells`) REFERENCES `sgo_cells` (`id`) ON
    UPDATE CASCADE,
  CONSTRAINT `fk_inc_cells_3` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_cells_BINS`
BEFORE INSERT ON `cco`.`inc_cells`
FOR EACH ROW
begin
        -- insert a new record in inc_divisions
        -- and get the new id for inc_cells
        insert into inc_divisions (id_division_type,id_incident) VALUES (3,NEW.id_incident);
        SET NEW.id = LAST_INSERT_ID();
        SET NEW.setting_time = NOW();
end

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_cells_AINS`
AFTER INSERT ON `cco`.`inc_cells`
FOR EACH ROW
begin
        -- depending on the incident phase change the structure of the SGO
        set @inc_phase = (select id_sgo_phase from incident where id = NEW.id_incident);
        case @inc_phase
        when 2 then             -- define 3 sectors
                insert into inc_sectors (id_sgo_sectors, id_incident) (select id, NEW.
    id_incident from sgo_sectors limit 3);
                insert into inc_cells_sectors (id_inc_cells, id_inc_sectors) (select NEW.id,
     id from inc_sectors where id_incident = NEW.id_incident);
        when 3 then             -- define 6 sectors
                insert into inc_sectors (id_sgo_sectors, id_incident) (select id, NEW.
    id_incident from sgo_sectors WHERE id NOT IN (SELECT id_sgo_sectors FROM inc_sectors
    WHERE id_incident = NEW.id_incident) LIMIT 6);
                insert into inc_cells_sectors (id_inc_cells, id_inc_sectors) (select NEW.id,
     id from inc_sectors WHERE id NOT IN (SELECT id_inc_sectors FROM inc_cells_sectors WHERE
     id_inc_cells = NEW.id));
        end case;
end
```

Code Excerpt C.9: *inc_cells*

```
CREATE TABLE `inc_cells_sectors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_inc_cells` int(11) NOT NULL,
  `id_inc_sectors` int(11) NOT NULL,
  `setting_time` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_cells_sectors_1_idx` (`id_inc_cells`),
  KEY `fk_inc_cells_sectors_2_idx` (`id_inc_sectors`),
  CONSTRAINT `fk_inc_cells_sectors_1` FOREIGN KEY (`id_inc_cells`) REFERENCES `inc_cells` (`
    id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_cells_sectors_2` FOREIGN KEY (`id_inc_sectors`) REFERENCES `inc_sectors
    ` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_cells_sectors_BINS`
BEFORE INSERT ON `cco`.`inc_cells_sectors`
FOR EACH ROW
begin
        SET NEW.setting_time = NOW();
end
```

Code Excerpt C.10: *inc_cells_sectors*

```
CREATE TABLE `inc_assistants` (
  `id` int(11) NOT NULL,
  `id_sgo_assistants` int(11) NOT NULL,
  `id_incident` int(11) NOT NULL,
  `setting_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_assistants_1_idx` (`id_sgo_assistants`),
  KEY `fk_inc_assistants_2_idx` (`id_incident`),
  CONSTRAINT `fk_inc_assistants_1` FOREIGN KEY (`id_sgo_assistants`) REFERENCES `
    sgo_assistants` (`id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_assistants_2` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_assistants_BINS`
BEFORE INSERT ON `cco`.`inc_assistants`
FOR EACH ROW

begin
        insert into inc_divisions (id_division_type,id_incident) VALUES (5,NEW.id_incident);
        SET NEW.id = LAST_INSERT_ID();
        SET NEW.setting_time = NOW();
end
```

Code Excerpt C.11: *inc_assistants*

```
CREATE TABLE `inc_zcr` (
  `id` int(11) NOT NULL,
  `id_inc_cells` int(11) NOT NULL,
  `inc_zcr_num` int(11) NOT NULL DEFAULT '1',
  `setting_time` datetime NOT NULL,
  `id_incident` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_zcr_2_idx` (`id_inc_cells`),
  KEY `fk_inc_zcr_3_idx` (`inc_zcr_num`),
  KEY `fk_inc_zcr_4_idx` (`id_incident`),
  CONSTRAINT `fk_inc_zcr_1` FOREIGN KEY (`id`) REFERENCES `inc_divisions` (`id`) ON DELETE
    CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_zcr_2` FOREIGN KEY (`id_inc_cells`) REFERENCES `inc_cells` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_zcr_3` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_zcr_BINS`
BEFORE INSERT ON `cco`.`inc_zcr`
FOR EACH ROW
begin
        insert into inc_divisions (id_division_type,id_incident) VALUES (4,NEW.id_incident);
        SET NEW.id = LAST_INSERT_ID();
        SET NEW.setting_time = NOW();
end
```

```
$$

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_zcr_AINS`
AFTER INSERT ON `cco`.`inc_zcr`
FOR EACH ROW
begin
        -- insert ZCR areas according to phase
        SET @inc_phase = (SELECT id_sgo_phase FROM incident WHERE id = NEW.id_incident);
        IF @inc_phase=3 THEN
                insert into inc_zcr_areas (id_inc_zcr,id_sgo_zcr) SELECT NEW.id, id FROM
    sgo_zcr WHERE id <3 OR id>6;
        END IF;
        IF @inc_phase=4 THEN
                insert into inc_zcr_areas (id_inc_zcr,id_sgo_zcr) SELECT NEW.id, id FROM
    sgo_zcr;
        END IF;
end
```

Code Excerpt C.12: *inc_zcr*

```
CREATE TABLE `sgo_zcr` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Code Excerpt C.13: *sgo_zcr*

```
CREATE TABLE `inc_zcr_areas` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_inc_zcr` int(11) NOT NULL,
  `id_sgo_zcr` int(11) NOT NULL,
  `setting_time` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_zcr_areas_1_idx` (`id_sgo_zcr`),
  KEY `fk_inc_zcr_areas_2_idx` (`id_inc_zcr`),
  CONSTRAINT `fk_inc_zcr_areas_1` FOREIGN KEY (`id_sgo_zcr`) REFERENCES `sgo_zcr` (`id`) ON
    UPDATE CASCADE,
  CONSTRAINT `fk_inc_zcr_areas_2` FOREIGN KEY (`id_inc_zcr`) REFERENCES `inc_zcr` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_zcr_areas_BINS`
BEFORE INSERT ON `cco`.`inc_zcr_areas`
FOR EACH ROW
begin
        SET NEW.setting_time = NOW();
end
```

Code Excerpt C.14: *inc_zcr_areas*

C.2.3 Incident occurrences cluster

```
CREATE TABLE `inc_damages` (
  `id` int(11) NOT NULL,
  `design` varchar(45) DEFAULT NULL,
  `affected_structure` varchar(45) DEFAULT NULL,
  `id_incident` int(11) NOT NULL,
  `id_inc_resources` int(11) NOT NULL,
  `comm_time` datetime NOT NULL,
  `observ` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_damage_BINS`
BEFORE INSERT ON `cco`.`inc_damages`
FOR EACH ROW
begin
        insert into inc_occurrence (`setting_time`,`id_type_occurrence`,`id_inc_division`)
    VALUES (NOW(),5, NEW.id_inc_resources);
        set NEW.id = LAST_INSERT_ID();
end
```

Code Excerpt C.15: *inc_damages*

```
CREATE TABLE `tb_victim_type` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` varchar(60) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1 COMMENT='Victim type'
```

Code Excerpt C.16: *tb_victim_type*

```
CREATE TABLE `tb_victim_state` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(45) NOT NULL DEFAULT 'NULL',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1 COMMENT='Victim state'
```

Code Excerpt C.17: *tb_victim_state*

```
CREATE TABLE `tb_victim_situation` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(45) NOT NULL DEFAULT 'NULL',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1 COMMENT='Victim situation'
```

Code Excerpt C.18: *tb_victim_situation*

```
CREATE TABLE `tb_victim_condition` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(45) NOT NULL DEFAULT 'NULL',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1 COMMENT='Victim condition'
```

Code Excerpt C.19: *tb_victim_condition*

```
CREATE TABLE `tb_nationalities` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) DEFAULT NULL,
  `abrev` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1
```

Code Excerpt C.20: *tb_nationalities*

```
CREATE TABLE `tb_fire_class` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) DEFAULT NULL,
  `info` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1
```

Code Excerpt C.21: *tb_fire_class*

```
CREATE TABLE `tb_fire_status` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1
```

Code Excerpt C.22: *tb_fire_status*

C.2.4 Incident resources cluster

```
CREATE TABLE `inc_resources` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `mob_time` datetime NOT NULL,
  `arriv_time` datetime DEFAULT NULL,
  `demob_time` datetime DEFAULT NULL,
  `first_responder` tinyint(1) NOT NULL DEFAULT '0',
  `id_type_resource` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Code Excerpt C.23: *inc_resources*

```
CREATE TABLE `inc_teams` (
  `id` int(11) NOT NULL,
  `num_team` int(11) NOT NULL DEFAULT '1',
  `setting_time` datetime NOT NULL,
  `id_incident` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_teams_3_idx` (`id_incident`),
  CONSTRAINT `fk_inc_teams_3` FOREIGN KEY (`id_incident`) REFERENCES `incident` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_teams_before_insert`
BEFORE INSERT ON `cco`.`inc_teams`
FOR EACH ROW
begin
        insert into inc_divisions (id_division_type,id_incident) VALUES (1,NEW.id_incident);
        SET NEW.id = LAST_INSERT_ID();
        SET NEW.setting_time = NOW();
end
```

Code Excerpt C.24: *inc_teams*

```
CREATE TABLE `inc_vehicles` (
  `id` int(11) NOT NULL,
  `id_corporation` int(11) NOT NULL,
  `weight` decimal(10,2) DEFAULT NULL,
  `autonomy` decimal(10,2) DEFAULT NULL,
  `id_tb_vehicle_type` int(11) NOT NULL,
  `capacity` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_vehicles_1_idx` (`id_tb_vehicle_type`),
  KEY `fk_inc_vehicles_2_idx` (`id_corporation`),
  CONSTRAINT `fk_inc_vehicles_1` FOREIGN KEY (`id_tb_vehicle_type`) REFERENCES `
    tb_vehicle_type` (`id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_vehicles_2` FOREIGN KEY (`id_corporation`) REFERENCES `tb_corporation`
    (`id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_vehicles_3` FOREIGN KEY (`id`) REFERENCES `inc_resources` (`id`) ON
    DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_vehicles_before_insert`
BEFORE INSERT ON `cco`.`inc_vehicles`
FOR EACH ROW
begin
        insert into inc_resources (`mob_time`) VALUES (NOW());
        set NEW.id = LAST_INSERT_ID();
end
```

Code Excerpt C.25: *inc_vehicles*

```
CREATE TABLE `inc_robots` (
  `id` int(11) NOT NULL,
  `id_corporation` int(11) NOT NULL,
  `id_tb_robot_class` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_robots_2_idx` (`id_tb_robot_class`),
  KEY `fk_inc_robots_3_idx` (`id_corporation`),
  CONSTRAINT `fk_inc_robots_1` FOREIGN KEY (`id`) REFERENCES `inc_resources` (`id`) ON
    UPDATE CASCADE,
  CONSTRAINT `fk_inc_robots_2` FOREIGN KEY (`id_tb_robot_class`) REFERENCES `tb_robot_class`
     (`id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_robots_3` FOREIGN KEY (`id_corporation`) REFERENCES `tb_corporation` (`
    id`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`inc_robots_before_insert`
BEFORE INSERT ON `cco`.`inc_robots`
FOR EACH ROW
begin
        insert into inc_resources (`mobtime`) VALUES (NOW());
        set NEW.id = LAST_INSERT_ID();
end
```

Code Excerpt C.26: *inc_robots*

```
CREATE TABLE `inc_vehicle_team` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_inc_vehicle` int(11) NOT NULL,
  `id_inc_teams` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_inc_vehicle_team_1_idx` (`id_inc_teams`),
  KEY `fk_inc_vehicle_team_2_idx` (`id_inc_vehicle`),
  CONSTRAINT `fk_inc_vehicle_team_1` FOREIGN KEY (`id_inc_teams`) REFERENCES `inc_teams` (`
    id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_inc_vehicle_team_2` FOREIGN KEY (`id_inc_vehicle`) REFERENCES `inc_vehicles
    ` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Code Excerpt C.27: *inc_vehicle_team*

```
CREATE TABLE `tb_vehicle_type` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) NOT NULL,
  `nick` varchar(10) NOT NULL,
  `info` text NOT NULL,
  `id_vehicle_group` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_tb_vehicle_type_1_idx` (`id_vehicle_group`),
  CONSTRAINT `fk_tb_vehicle_type_1` FOREIGN KEY (`id_vehicle_group`) REFERENCES `
    tb_vehicle_group` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=latin1
```

Code Excerpt C.28: *tb_vehicle_type*

```
CREATE TABLE `tb_vehicle_group` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) NOT NULL,
  `info` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=latin1
```

Code Excerpt C.29: *tb_vehicle_group*

```
CREATE TABLE `tb_robot_class` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) NOT NULL,
  `photo` blob,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Code Excerpt C.30: *tb_robot_class*

C.2.5 Incident personnel cluster

```
CREATE TABLE `tb_person` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `surname` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `id_entity` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_tb_person_1_idx` (`id_entity`),
  CONSTRAINT `fk_tb_person_1` FOREIGN KEY (`id_entity`) REFERENCES `tb_entities` (`id`) ON
    DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=42 DEFAULT CHARSET=latin1
```

Code Excerpt C.31: *tb_person*

```
CREATE TABLE `tb_entities` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(60) DEFAULT NULL,
  `abrev` varchar(10) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=latin1
```

Code Excerpt C.32: *tb_entities*

```
CREATE TABLE `tb_corporation` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(60) DEFAULT NULL,
  `id_tb_conc` int(11) DEFAULT NULL,
  `id_tb_dist` int(11) DEFAULT NULL,
  `numero` varchar(4) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `id_tb_conc` (`id_tb_conc`),
  KEY `id_tb_dist` (`id_tb_dist`),
  CONSTRAINT `tb_corporation_ibfk_2` FOREIGN KEY (`id_tb_conc`) REFERENCES `tb_conc` (`id`),
  CONSTRAINT `tb_corporation_ibfk_3` FOREIGN KEY (`id_tb_dist`) REFERENCES `tb_dist` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=450 DEFAULT CHARSET=latin1
```

Code Excerpt C.33: *tb_corporation*

```
CREATE TABLE `tb_firefighter` (
  `id` int(11) NOT NULL,
  `id_corporation` int(11) NOT NULL,
  `id_rank` int(11) NOT NULL,
  `num_mec` int(11) NOT NULL,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `age` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_tb_firefighter_2_idx` (`id_rank`),
  KEY `fk_tb_firefighter_3_idx` (`id_corporation`),
  CONSTRAINT `fk_tb_firefighter_1` FOREIGN KEY (`id`) REFERENCES `tb_person` (`id`) ON
    DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_tb_firefighter_2` FOREIGN KEY (`id_rank`) REFERENCES `tb_firefighter_rank`
    (`id`) ON UPDATE CASCADE,
  CONSTRAINT `fk_tb_firefighter_3` FOREIGN KEY (`id_corporation`) REFERENCES `tb_corporation
    ` (`id`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE
DEFINER=`root`@`localhost`
TRIGGER `cco`.`tb_firefighter_BINS`
BEFORE INSERT ON `cco`.`tb_firefighter`
FOR EACH ROW
begin
        insert into tb_person (`surname`,`name`,`id_entity`) values (NEW.`surname`,NEW.`name
    `,1);
        SET NEW.id = LAST_INSERT_ID();
end
```

Code Excerpt C.34: *tb_firefighter*

```
CREATE TABLE `tb_firefighter_rank` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `rank` varchar(60) NOT NULL,
  `rank_image` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=latin1
```

Code Excerpt C.35: *tb_firefighter_rank*

C.2.6 Incident location cluster

```
CREATE TABLE `tb_dist` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=latin1
```

Code Excerpt C.36: *tb_dist*

```
CREATE TABLE `tb_conc` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) DEFAULT NULL,
  `id_tb_dist` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `id_tb_dist` (`id_tb_dist`),
  CONSTRAINT `tb_conc_ibfk_1` FOREIGN KEY (`id_tb_dist`) REFERENCES `tb_dist` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=310 DEFAULT CHARSET=latin1
```

Code Excerpt C.37: *tb_conc*

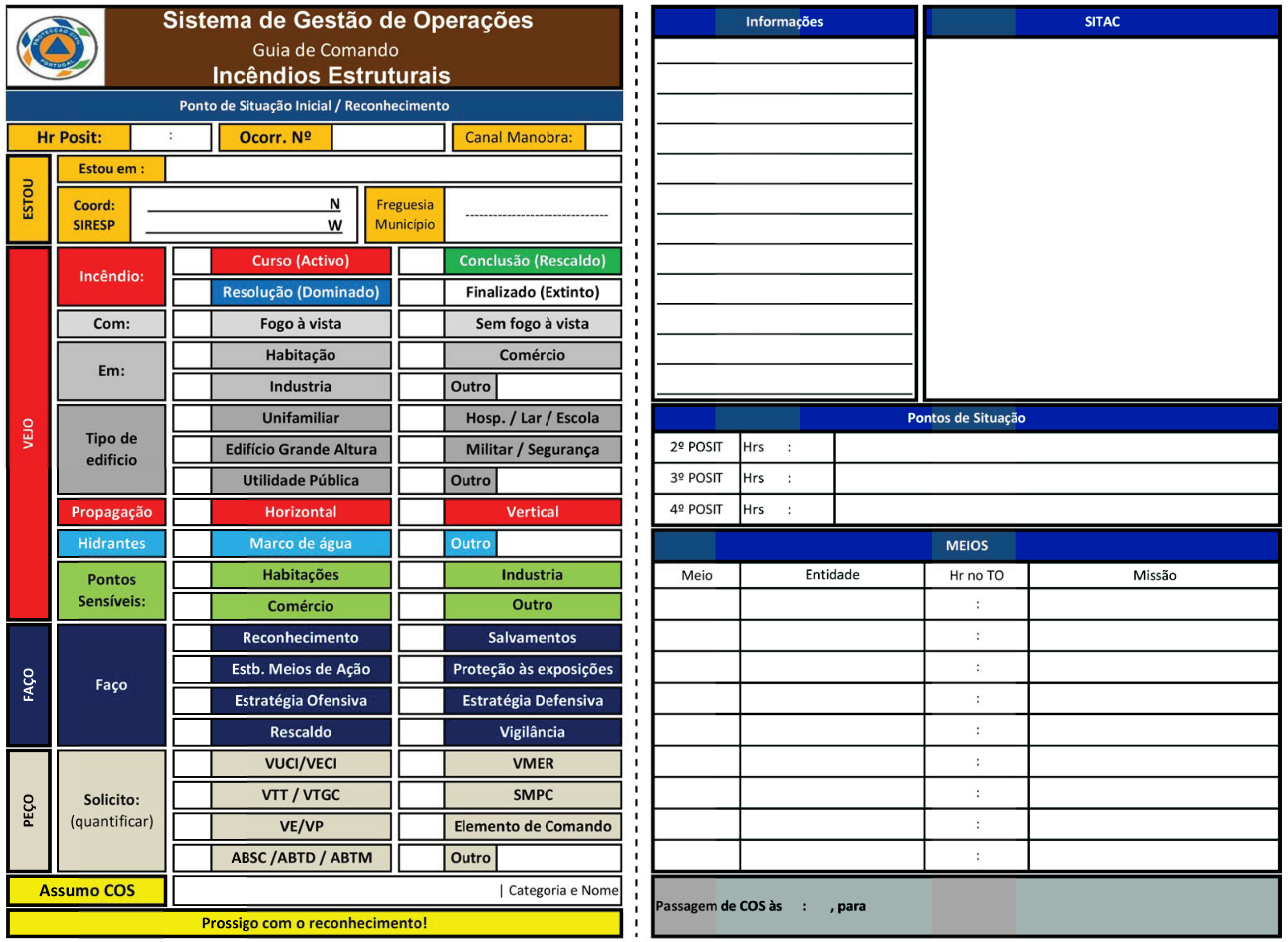
```
CREATE TABLE `tb_freg` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `design` varchar(60) DEFAULT NULL,
  `id_tb_conc` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `id_tb_conc` (`id_tb_conc`),
  CONSTRAINT `tb_freg_ibfk_1` FOREIGN KEY (`id_tb_conc`) REFERENCES `tb_conc` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4281 DEFAULT CHARSET=latin1
```

Code Excerpt C.38: *tb_freg*

Appendix D

ANPC forms

D.1 CECOP forms



Sistema de Gestão de Operações
Guia de Comando
Incêndios Estruturais

Ponto de Situação Inicial / Reconhecimento

Hr Posit:	:	**Ocorr. Nº**		Canal Manobra:	

ESTOU	**Estou em :**				
	Coord: SIRESP	N W	Freguesia Município	----------------------------	
VEJO	**Incêndio:**		**Curso (Activo)**		**Conclusão (Rescaldo)**
			Resolução (Dominado)		**Finalizado (Extinto)**
	Com:		**Fogo à vista**		**Sem fogo à vista**
	Em:		**Habitação**		**Comércio**
			Industria		**Outro**
	Tipo de edificio		**Unifamiliar**		**Hosp. / Lar / Escola**
			Edifício Grande Altura		**Militar / Segurança**
			Utilidade Pública		**Outro**
	Propagação		**Horizontal**		**Vertical**
	Hidrantes		**Marco de água**		**Outro**
	Pontos Sensíveis:		**Habitações**		**Industria**
			Comércio		**Outro**
FAÇO	**Faço**		**Reconhecimento**		**Salvamentos**
			Estb. Meios de Ação		**Proteção às exposições**
			Estratégia Ofensiva		**Estratégia Defensiva**
			Rescaldo		**Vigilância**
PEÇO	**Solicito:** (quantificar)		**VUCI/VECI**		**VMER**
			VTT / VTGC		**SMPC**
			VE/VP		**Elemento de Comando**
			ABSC /ABTD / ABTM		**Outro**

Assumo COS	| Categoria e Nome

Prossigo com o reconhecimento!

Informações

SITAC

Pontos de Situação

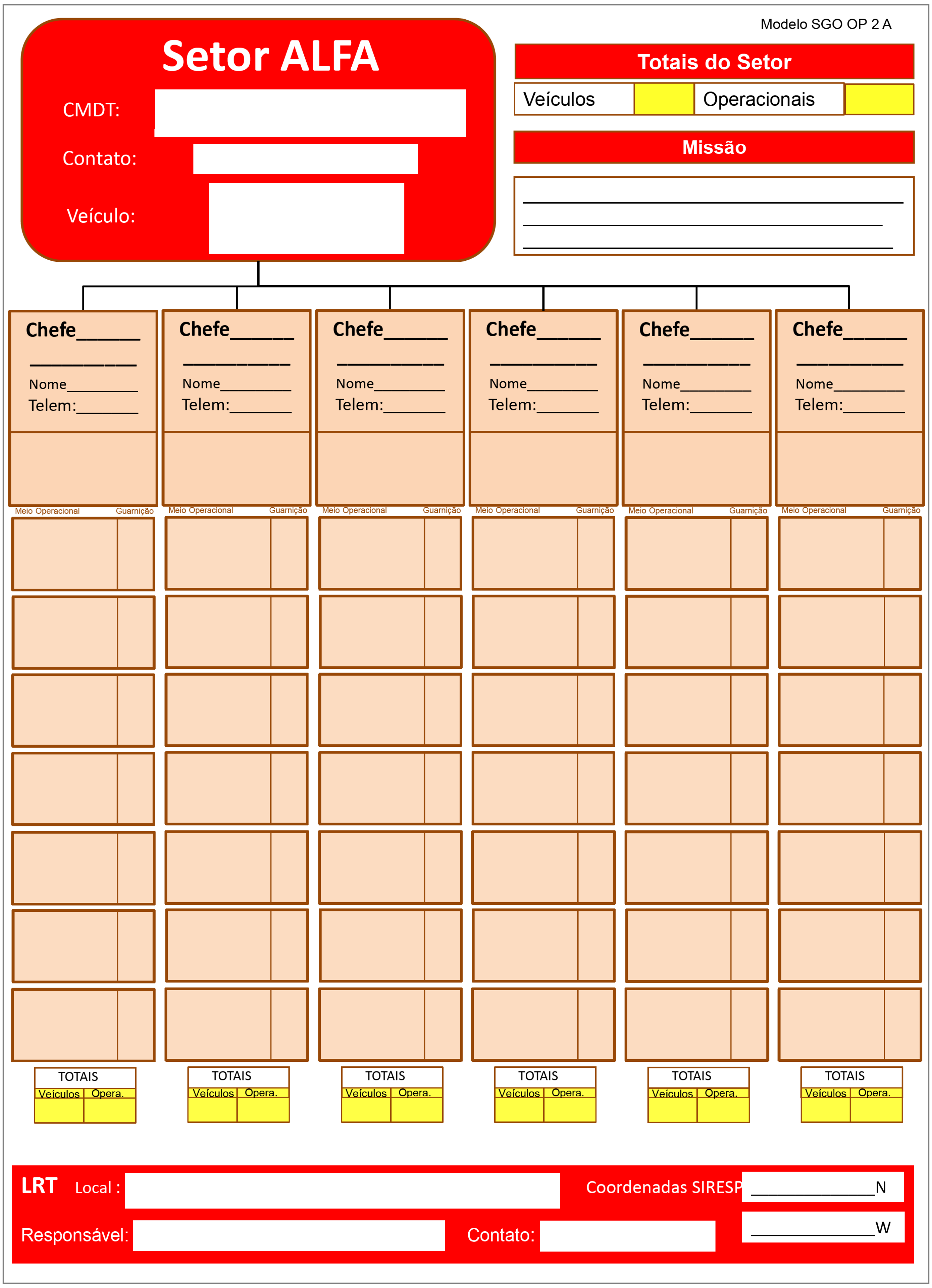
2º POSIT	Hrs :	
3º POSIT	Hrs :	
4º POSIT	Hrs :	

MEIOS

Meio	Entidade	Hr no TO	Missão
		:	
		:	
		:	
		:	
		:	
		:	
		:	
		:	
		:	

Passagem **de COS às : , para**

Figure D.1: Structural incidents command guide form.



Modelo SGO OP 2 A

Setor ALFA

CMDT:

Contato:

Veículo:

Totais do Setor

Veículos		Operacionais	

Missão

____

____

____

Chefe______	Chefe______	Chefe______	Chefe______	Chefe______	Chefe______
__________	__________	__________	__________	__________	__________
Nome________	Nome________	Nome________	Nome________	Nome________	Nome________
Telem:_______	Telem:_______	Telem:_______	Telem:_______	Telem:_______	Telem:_______

Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição

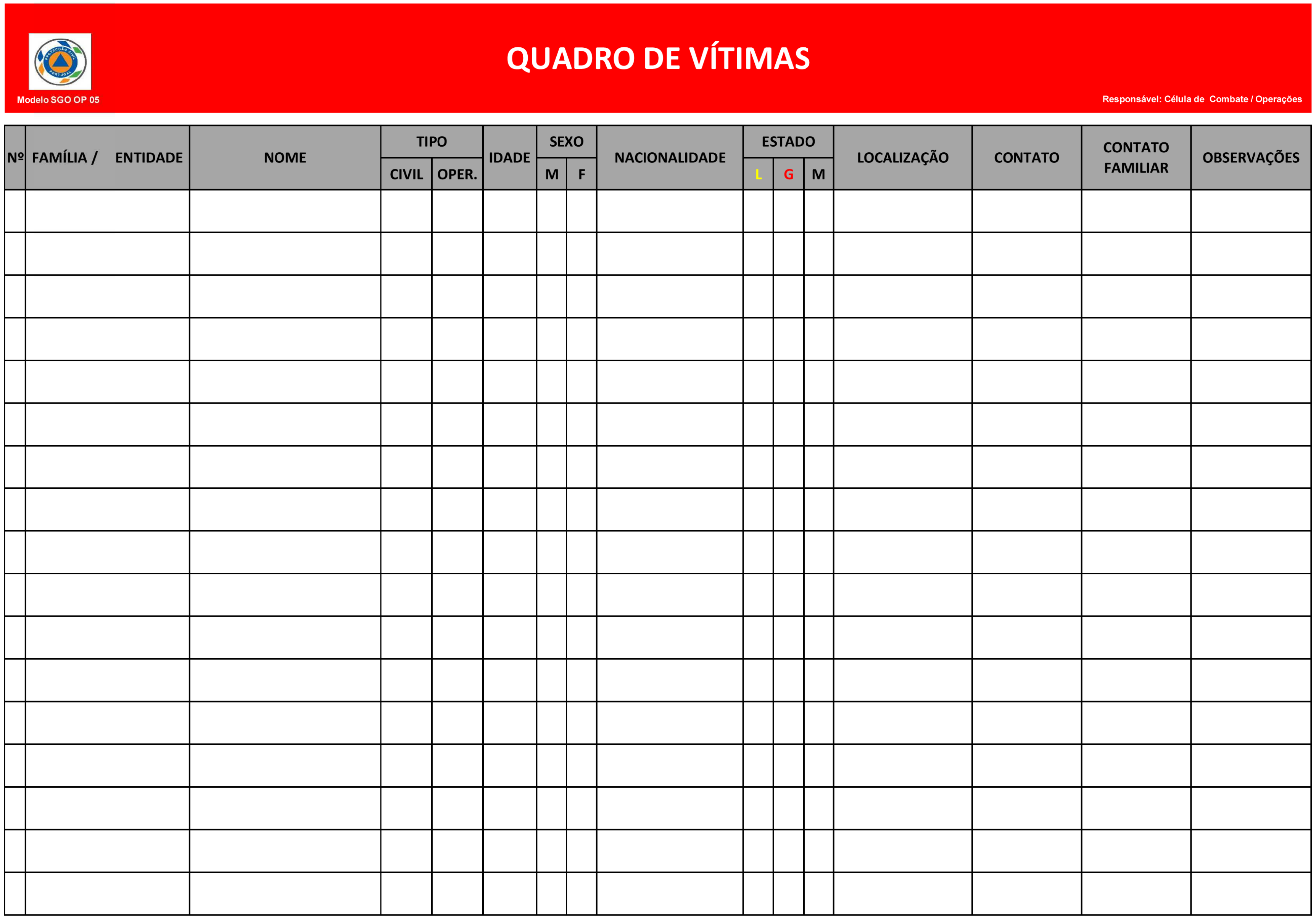
TOTAIS		TOTAIS		TOTAIS		TOTAIS		TOTAIS		TOTAIS	
Veículos	Opera.	Veículos	Opera.	Veículos	Opera.	Veículos	Opera.	Veículos	Opera.	Veículos	Opera.

LRT Local : Coordenadas SIRESP ______________N

______________W

Responsável: Contato:

FIGURE D.2: Sectorization form.



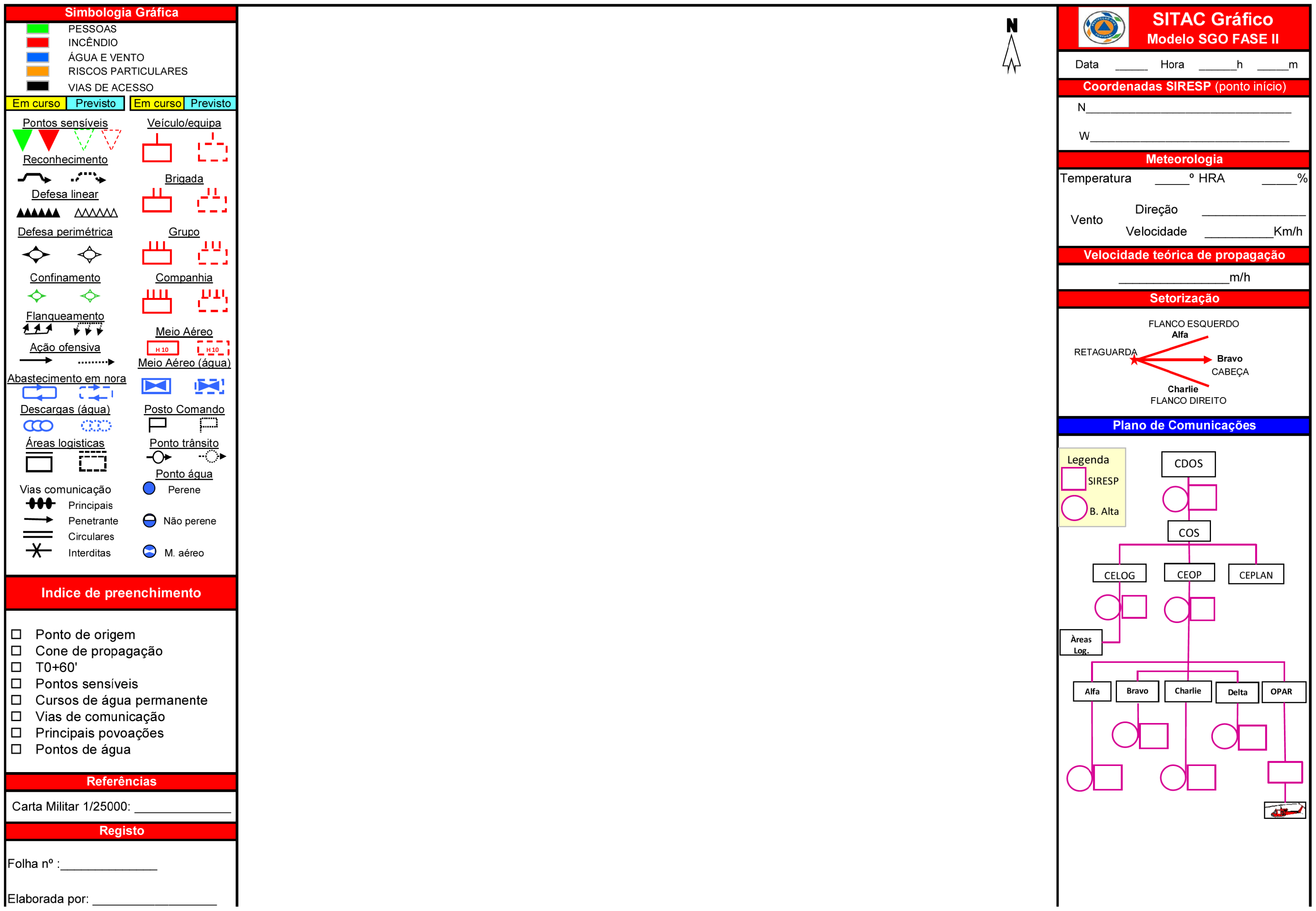
Modelo SGO OP 05

QUADRO DE VÍTIMAS

Responsável: Célula de Combate / Operações

Nº	FAMÍLIA / ENTIDADE	NOME	TIPO		IDADE	SEXO		NACIONALIDADE	ESTADO			LOCALIZAÇÃO	CONTATO	CONTATO FAMILIAR	OBSERVAÇÕES
			CIVIL	OPER.		M	F		L	G	M				

Figure D.3: Victims map form.



Simbologia Gráfica

🟩	PESSOAS
🟥	INCÊNDIO
🟦	ÁGUA E VENTO
🟧	RISCOS PARTICULARES
⬛	VIAS DE ACESSO

Em curso	Previsto	Em curso	Previsto

Pontos sensíveis; Reconhecimento; Defesa linear; Defesa perimétrica; Confinamento; Flanqueamento; Ação ofensiva; Abastecimento em nora; Descargas (água); Áreas logisticas; Vias comunicação: Principais, Penetrante, Circulares, Interditas

Veículo/equipa; Brigada; Grupo; Companhia; Meio Aéreo (H 10); Meio Aéreo (água); Posto Comando; Ponto trânsito; Ponto água: Perene, Não perene, M. aéreo

Indice de preenchimento

- ☐ Ponto de origem
- ☐ Cone de propagação
- ☐ T0+60'
- ☐ Pontos sensíveis
- ☐ Cursos de água permanente
- ☐ Vias de comunicação
- ☐ Principais povoações
- ☐ Pontos de água

Referências

Carta Militar 1/25000: ______________

Registo

Folha nº : ______________

Elaborada por: __________________

SITAC Gráfico
Modelo SGO FASE II

Data _____ Hora ______h _____m

Coordenadas SIRESP (ponto início)

N_______________________________

W_______________________________

Meteorologia

Temperatura _____° HRA _____%

Vento Direção _____________

Velocidade __________Km/h

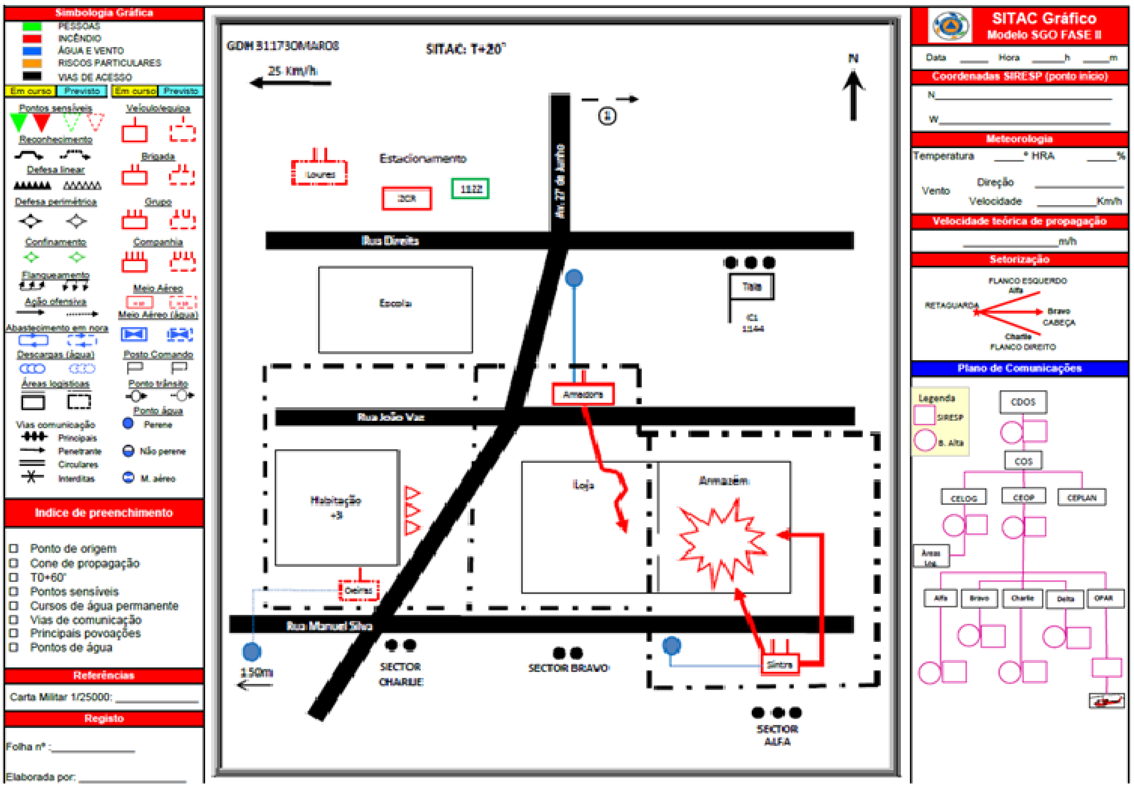
Velocidade teórica de propagação

_______________m/h

Setorização

Plano de Comunicações

FIGURE D.4: Graphical SITAC form.



Indice de preenchimento

- ☐ Ponto de origem
- ☐ Cone de propagação
- ☐ T0+60'
- ☐ Pontos sensíveis
- ☐ Cursos de água permanente
- ☐ Vias de comunicação
- ☐ Principais povoações
- ☐ Pontos de água

Referências

Carta Militar 1/25000: ____________

Registo

Folha nº : ____________

Elaborada por: ____________

Data _____ Hora _____h _____m

N ____________

W ____________

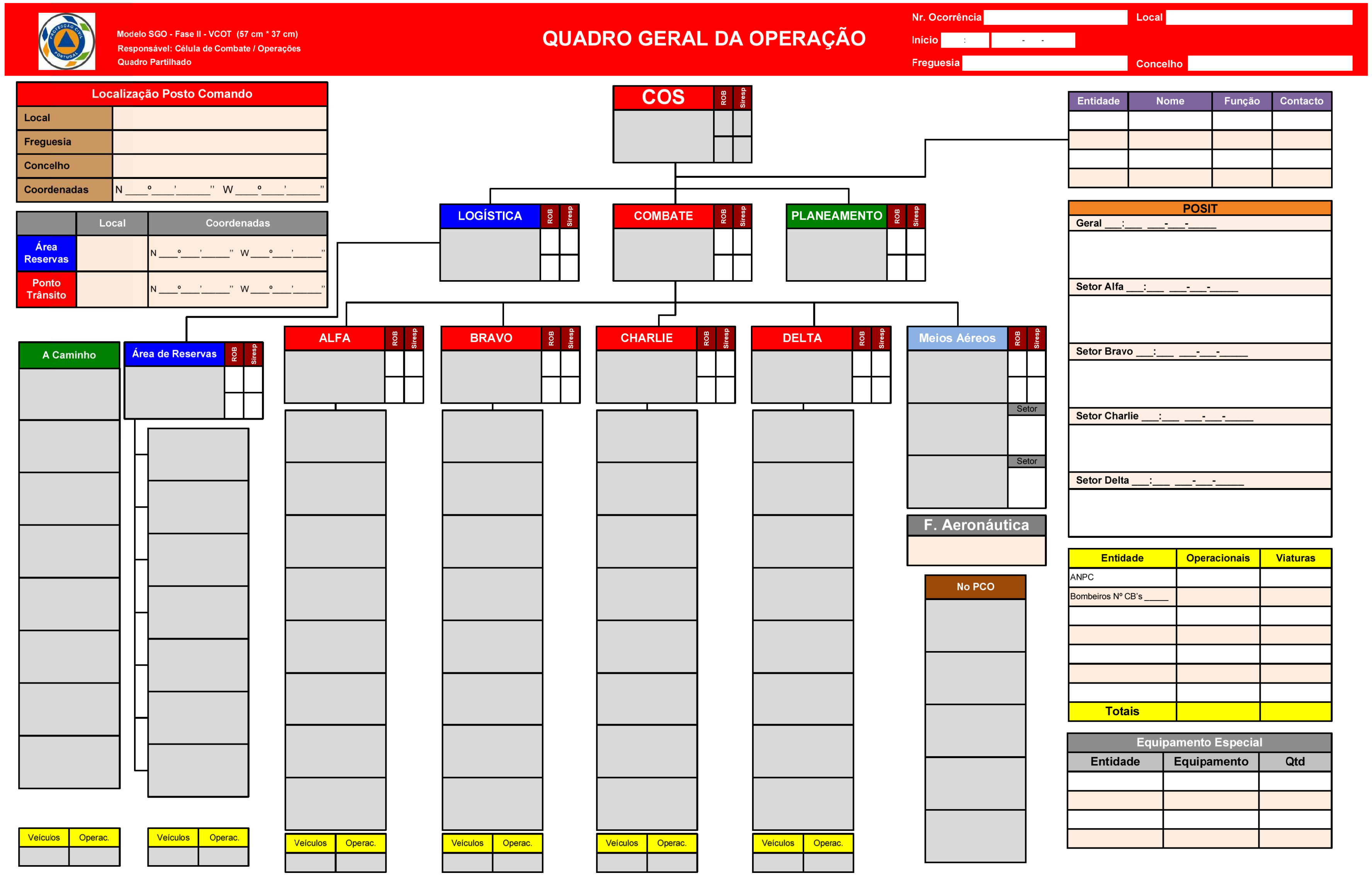
Temperatura _____º HRA _____%

Vento Direção ____________

Velocidade ____________Km/h

____________m/h

Figure D.5: Example of a graphical SITAC.



Modelo SGO - Fase II - VCOT (57 cm * 37 cm)
Responsável: Célula de Combate / Operações
Quadro Partilhado

QUADRO GERAL DA OPERAÇÃO

Nr. Ocorrência ____ Local ____
Início __ : __ __ - __ - __
Freguesia ____ Concelho ____

Localização Posto Comando	
Local	
Freguesia	
Concelho	
Coordenadas	N ___° ___' _____" W ___° ___' _____"

	Local	Coordenadas
Área Reservas		N ___° ___' _____" W ___° ___' _____"
Ponto Trânsito		N ___° ___' _____" W ___° ___' _____"

Entidade	Nome	Função	Contacto

COS ROB Siresp

LOGÍSTICA ROB Siresp

COMBATE ROB Siresp

PLANEAMENTO ROB Siresp

A Caminho

Área de Reservas ROB Siresp

ALFA ROB Siresp

BRAVO ROB Siresp

CHARLIE ROB Siresp

DELTA ROB Siresp

Meios Aéreos ROB Siresp

Setor

Setor

F. Aeronáutica

No PCO

Veículos Operac.

Veículos Operac.

Veículos Operac.

Veículos Operac.

Veículos Operac.

Veículos Operac.

POSIT

Geral __:__ __-__-____

Setor Alfa __:__ __-__-____

Setor Bravo __:__ __-__-____

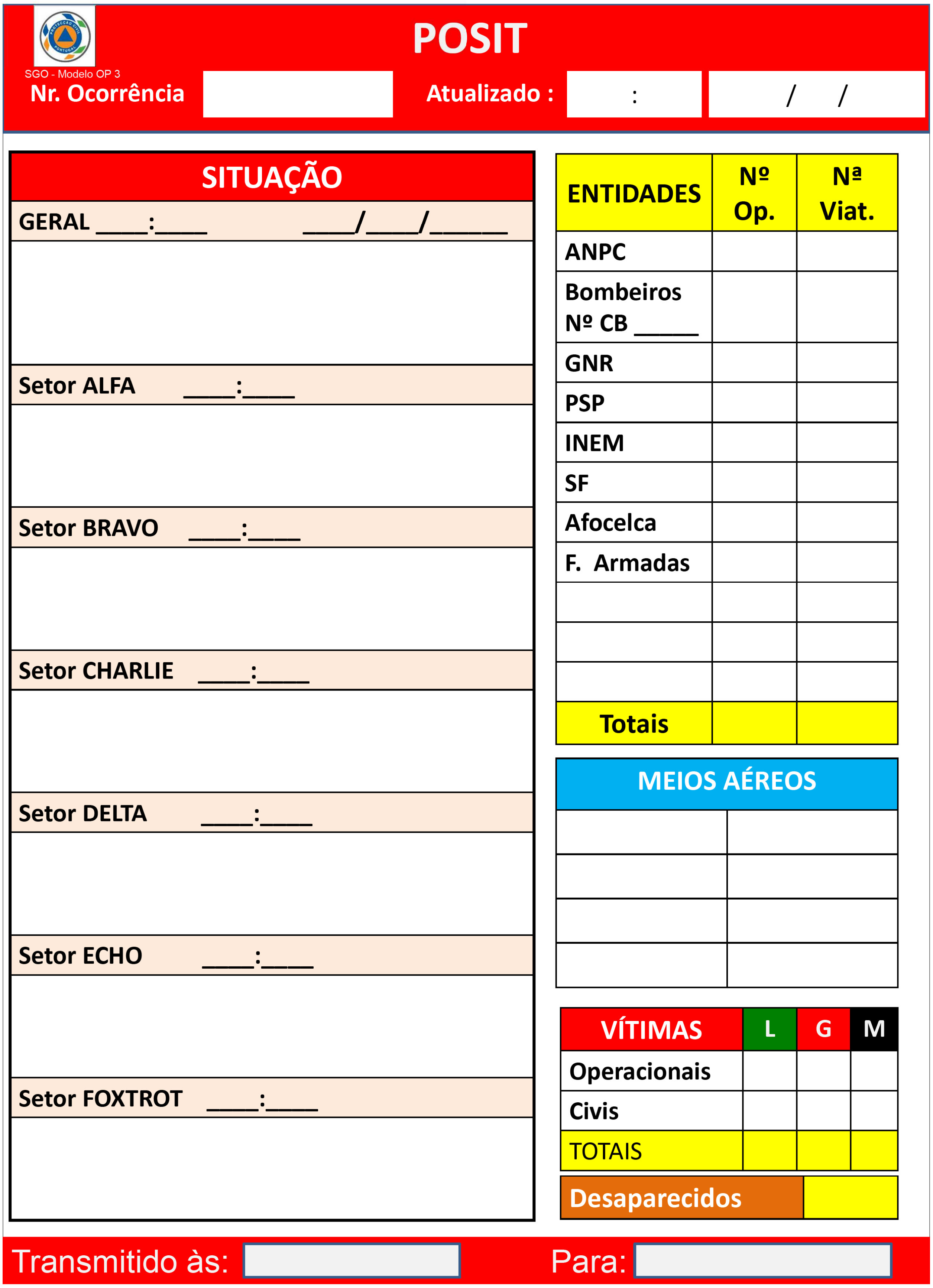
Setor Charlie __:__ __-__-____

Setor Delta __:__ __-__-____

Entidade	Operacionais	Viaturas
ANPC		
Bombeiros Nº CB's _____		
Totais		

Equipamento Especial		
Entidade	Equipamento	Qtd

Figure D.6: General framework of operations form.



SGO - Modelo OP 3

POSIT

Nr. Ocorrência ______ **Atualizado :** ___ : ___ ___ / ___ / ___

SITUAÇÃO
GERAL ____:____ ____/____/_____
Setor ALFA ____:____
Setor BRAVO ____:____
Setor CHARLIE ____:____
Setor DELTA ____:____
Setor ECHO ____:____
Setor FOXTROT ____:____

ENTIDADES	**Nº Op.**	**Nª Viat.**
ANPC		
Bombeiros Nº CB _____		
GNR		
PSP		
INEM		
SF		
Afocelca		
F. Armadas		
Totais		

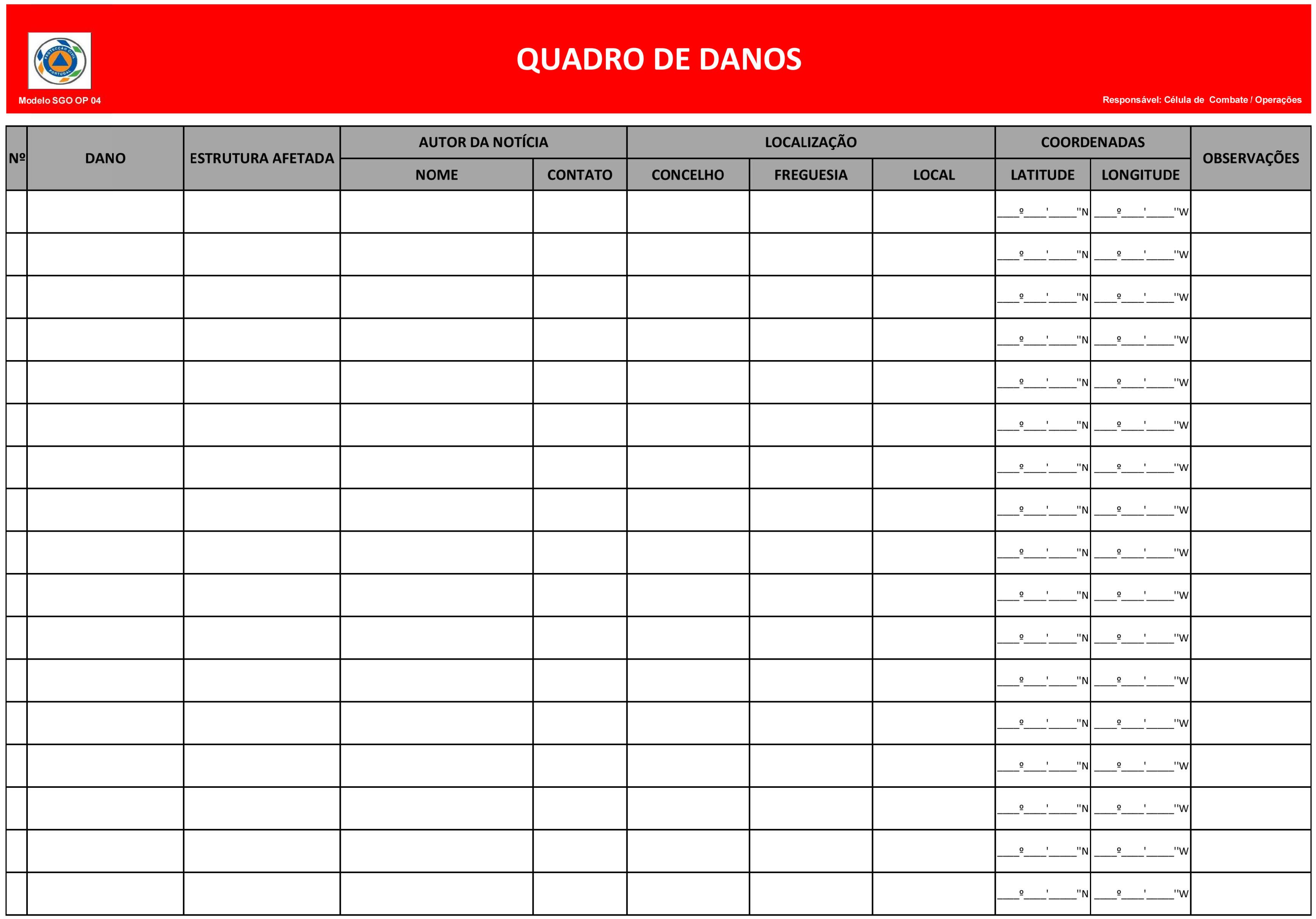
MEIOS AÉREOS	

VÍTIMAS	**L**	**G**	**M**
Operacionais			
Civis			
TOTAIS			

Desaparecidos	

Transmitido às: ______ Para: ______

FIGURE D.7: Current situation progress report.



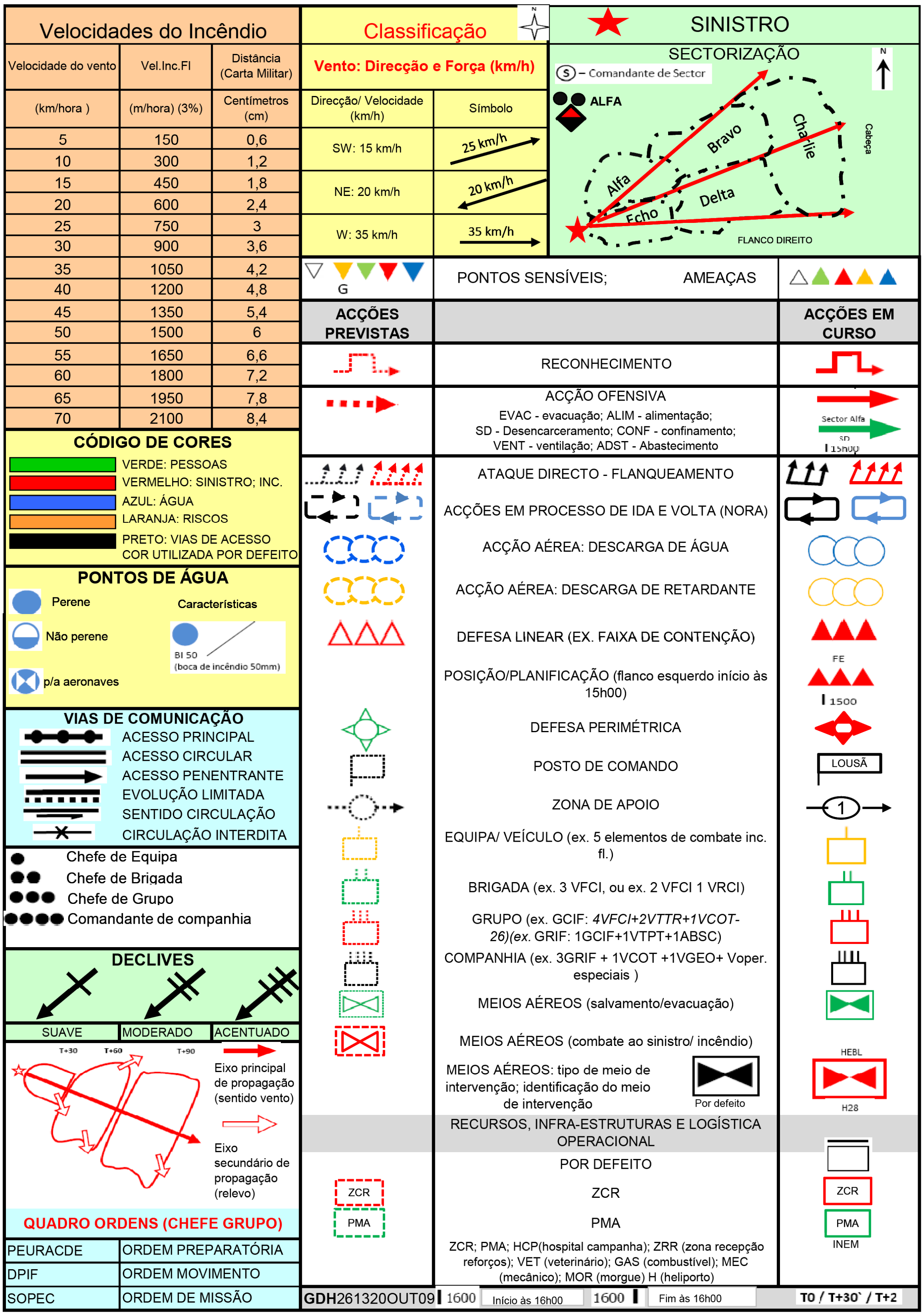
Modelo SGO OP 04

QUADRO DE DANOS

Responsável: Célula de Combate / Operações

Nº	DANO	ESTRUTURA AFETADA	AUTOR DA NOTÍCIA		LOCALIZAÇÃO			COORDENADAS		OBSERVAÇÕES
			NOME	CONTATO	CONCELHO	FREGUESIA	LOCAL	LATITUDE	LONGITUDE	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	
								___º___'____"N	___º___'____"W	

FIGURE D.8: Damage form.



Velocidades do Incêndio

Velocidade do vento	Vel.Inc.Fl	Distância (Carta Militar)
(km/hora)	(m/hora) (3%)	Centímetros (cm)
5	150	0,6
10	300	1,2
15	450	1,8
20	600	2,4
25	750	3
30	900	3,6
35	1050	4,2
40	1200	4,8
45	1350	5,4
50	1500	6
55	1650	6,6
60	1800	7,2
65	1950	7,8
70	2100	8,4

Classificação

Vento: Direcção e Força (km/h)

Direcção/ Velocidade (km/h)	Símbolo
SW: 15 km/h	25 km/h
NE: 20 km/h	20 km/h
W: 35 km/h	35 km/h

SINISTRO

SECTORIZAÇÃO

(S) – Comandante de Sector

ALFA

Bravo, Charlie, Alfa, Delta, Echo, Cabeça, FLANCO DIREITO, N

CÓDIGO DE CORES

🟩	VERDE: PESSOAS
🟥	VERMELHO: SINISTRO; INC.
🟦	AZUL: ÁGUA
🟧	LARANJA: RISCOS
⬛	PRETO: VIAS DE ACESSO COR UTILIZADA POR DEFEITO

PONTOS DE ÁGUA

Perene

Não perene

p/a aeronaves

Características: BI 50 (boca de incêndio 50mm)

VIAS DE COMUNICAÇÃO

ACESSO PRINCIPAL
ACESSO CIRCULAR
ACESSO PENENTRANTE
EVOLUÇÃO LIMITADA
SENTIDO CIRCULAÇÃO
CIRCULAÇÃO INTERDITA

● Chefe de Equipa
●● Chefe de Brigada
●●● Chefe de Grupo
●●●● Comandante de companhia

DECLIVES

SUAVE	MODERADO	ACENTUADO

T+30 T+60 T+90

Eixo principal de propagação (sentido vento)

Eixo secundário de propagação (relevo)

QUADRO ORDENS (CHEFE GRUPO)

PEURACDE	ORDEM PREPARATÓRIA
DPIF	ORDEM MOVIMENTO
SOPEC	ORDEM DE MISSÃO

G — PONTOS SENSÍVEIS; AMEAÇAS

ACÇÕES PREVISTAS		ACÇÕES EM CURSO
	RECONHECIMENTO	

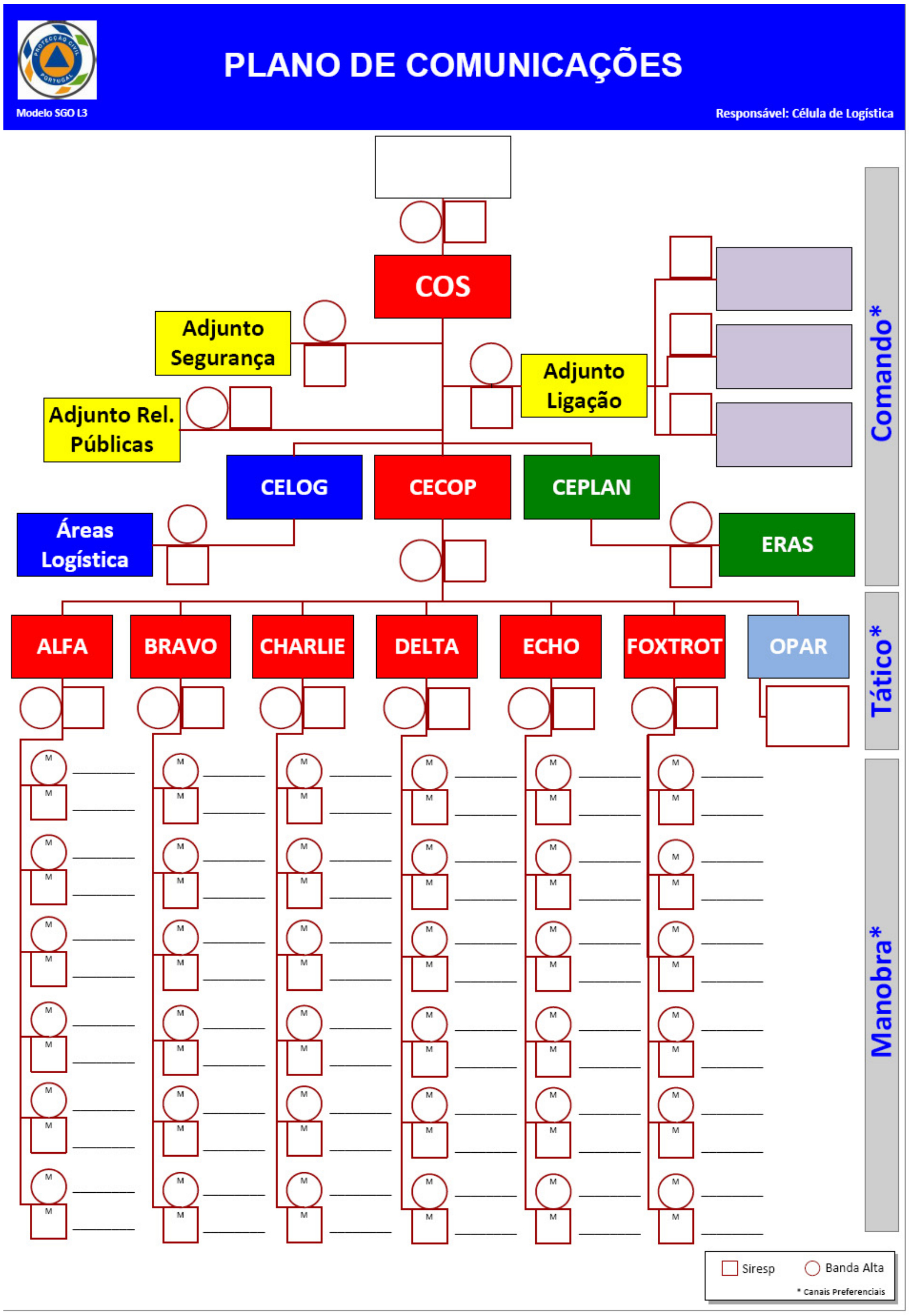
	ACÇÃO OFENSIVA
	EVAC - evacuação; ALIM - alimentação;
	SD - Desencarceramento; CONF - confinamento;
	VENT - ventilação; ADST - Abastecimento
		Sector Alfa
		SD
		15h00

	ATAQUE DIRECTO - FLANQUEAMENTO	
	ACÇÕES EM PROCESSO DE IDA E VOLTA (NORA)	
	ACÇÃO AÉREA: DESCARGA DE ÁGUA	
	ACÇÃO AÉREA: DESCARGA DE RETARDANTE	
	DEFESA LINEAR (EX. FAIXA DE CONTENÇÃO)	
	POSIÇÃO/PLANIFICAÇÃO (flanco esquerdo início às 15h00)	FE 1500
	DEFESA PERIMÉTRICA	
	POSTO DE COMANDO	LOUSÃ
	ZONA DE APOIO	1
	EQUIPA/ VEÍCULO (ex. 5 elementos de combate inc. fl.)	
	BRIGADA (ex. 3 VFCI, ou ex. 2 VFCI 1 VRCI)	
	GRUPO (ex. GCIF: *4VFCI+2VTTR+1VCOT-26)(ex.* GRIF: 1GCIF+1VTPT+1ABSC)	
	COMPANHIA (ex. 3GRIF + 1VCOT +1VGEO+ Voper. especiais)	
	MEIOS AÉREOS (salvamento/evacuação)	
	MEIOS AÉREOS (combate ao sinistro/ incêndio)	HEBL
	MEIOS AÉREOS: tipo de meio de intervenção; identificação do meio de intervenção — Por defeito	H28
	RECURSOS, INFRA-ESTRUTURAS E LOGÍSTICA OPERACIONAL	
	POR DEFEITO	
ZCR	ZCR	ZCR
PMA	PMA	PMA INEM
	ZCR; PMA; HCP(hospital campanha); ZRR (zona recepção reforços); VET (veterinário); GAS (combustível); MEC (mecânico); MOR (morgue) H (heliporto)	

GDH261320OUT09 | 1600 Início às 16h00 | 1600 Fim às 16h00 | T0 / T+30` / T+2

Figure D.9: Symbology for graphical SITAC.

D.2 CELOG forms



PLANO DE COMUNICAÇÕES

Modelo SGO L3

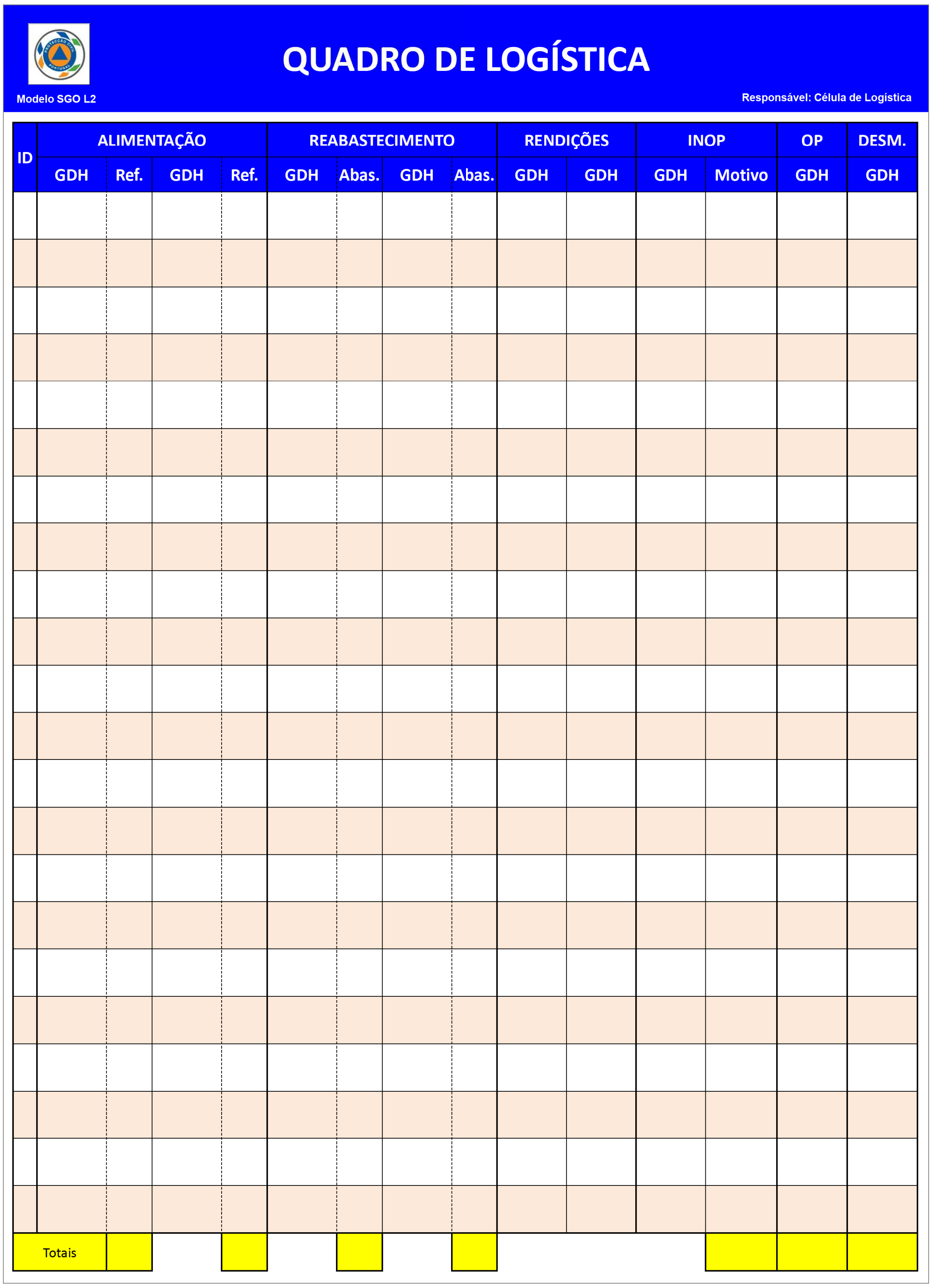
Responsável: Célula de Logística

COS
Adjunto Segurança
Adjunto Ligação
Adjunto Rel. Públicas
CELOG
CECOP
CEPLAN
Áreas Logística
ERAS
ALFA
BRAVO
CHARLIE
DELTA
ECHO
FOXTROT
OPAR
M
Comando*
Tático*
Manobra*

☐ Siresp ◯ Banda Alta

* Canais Preferenciais

FIGURE D.10: Communications plan form.



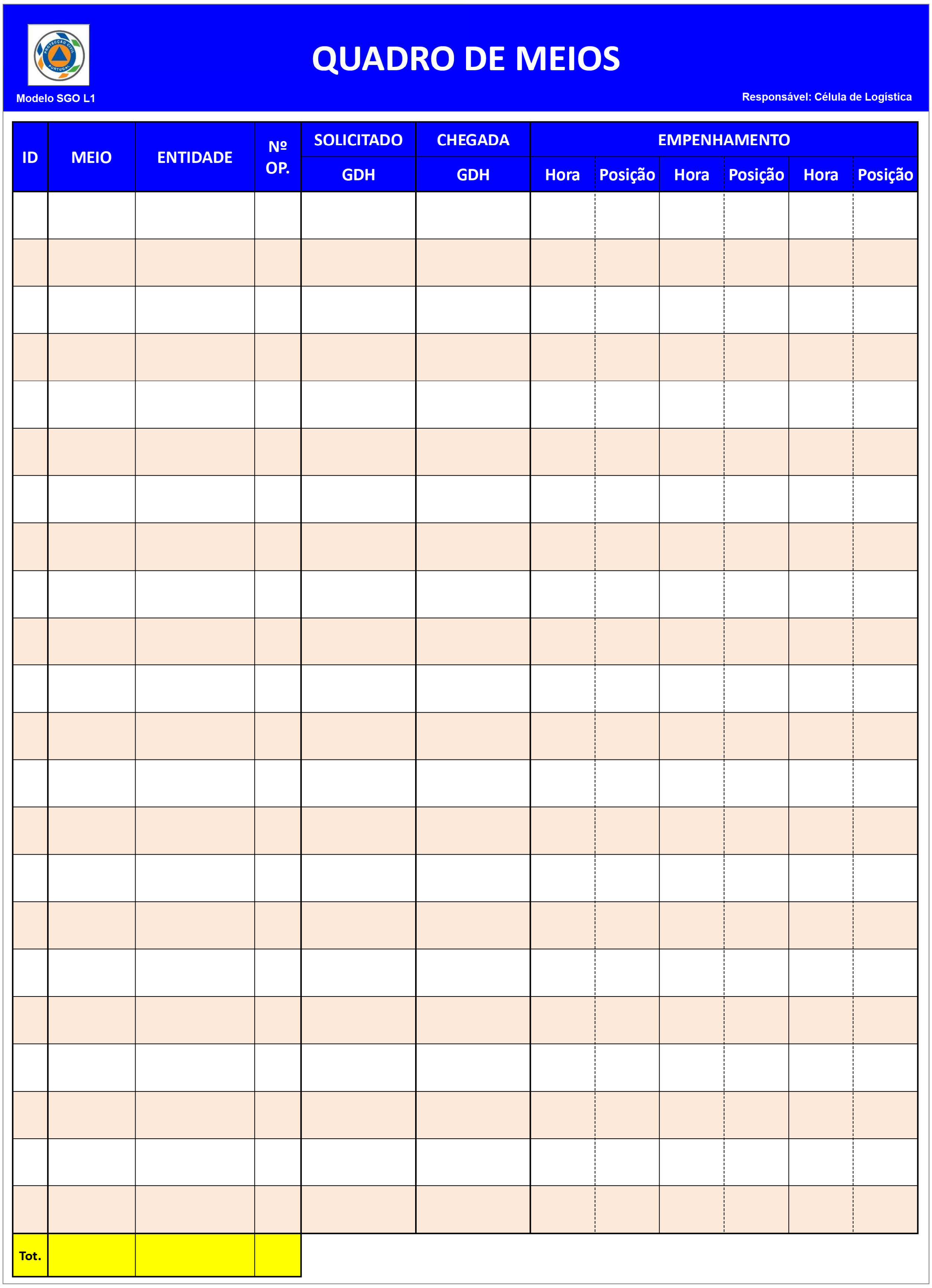
Modelo SGO L2

QUADRO DE LOGÍSTICA

Responsável: Célula de Logística

ID	ALIMENTAÇÃO				REABASTECIMENTO				RENDIÇÕES		INOP		OP	DESM.
	GDH	Ref.	GDH	Ref.	GDH	Abas.	GDH	Abas.	GDH	GDH	GDH	Motivo	GDH	GDH
Totais														

FIGURE D.11: Logistics form.



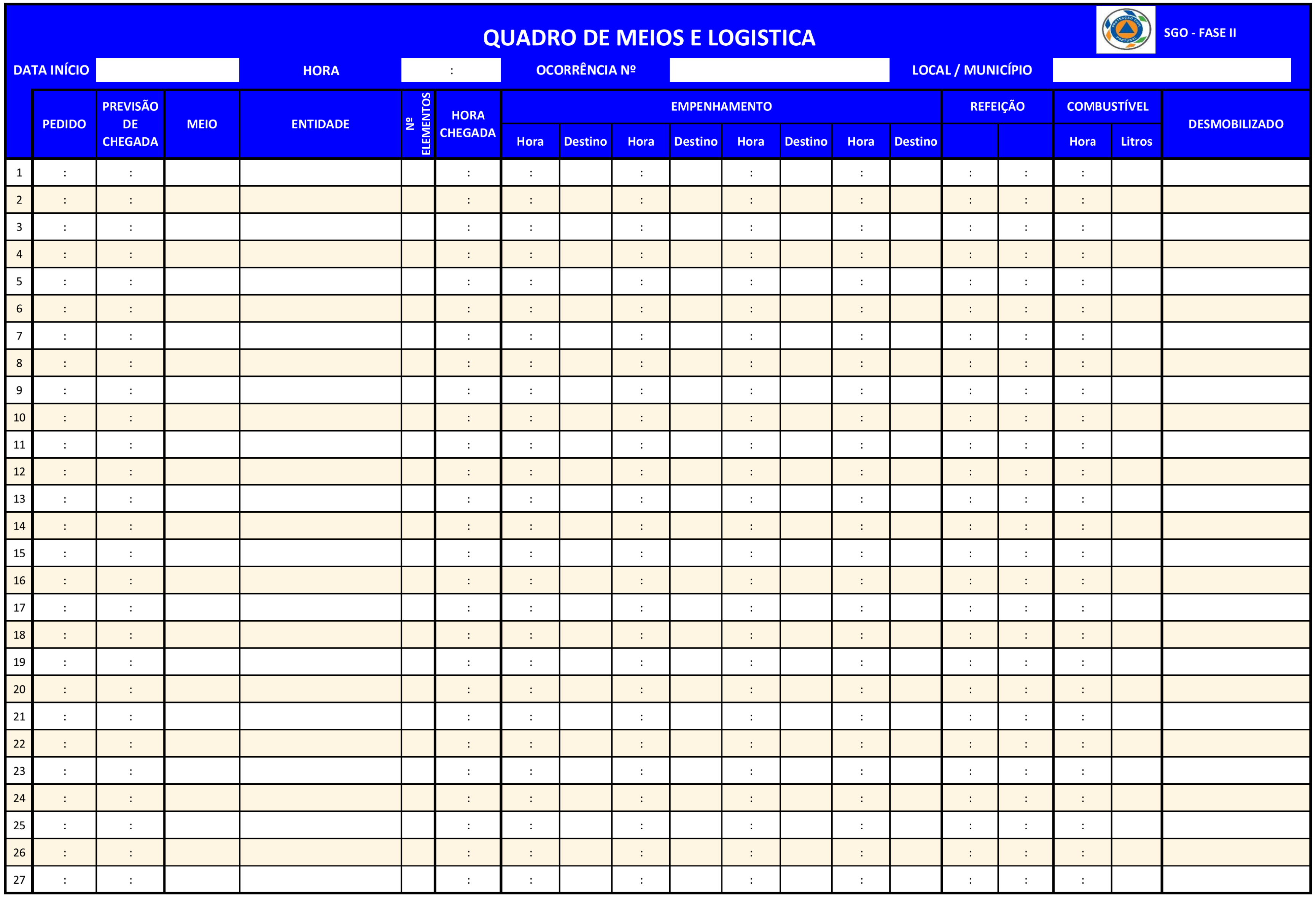
QUADRO DE MEIOS

Modelo SGO L1

Responsável: Célula de Logística

ID	MEIO	ENTIDADE	Nº OP.	SOLICITADO	CHEGADA	EMPENHAMENTO					
				GDH	GDH	Hora	Posição	Hora	Posição	Hora	Posição
Tot.											

Figure D.12: Resources form.



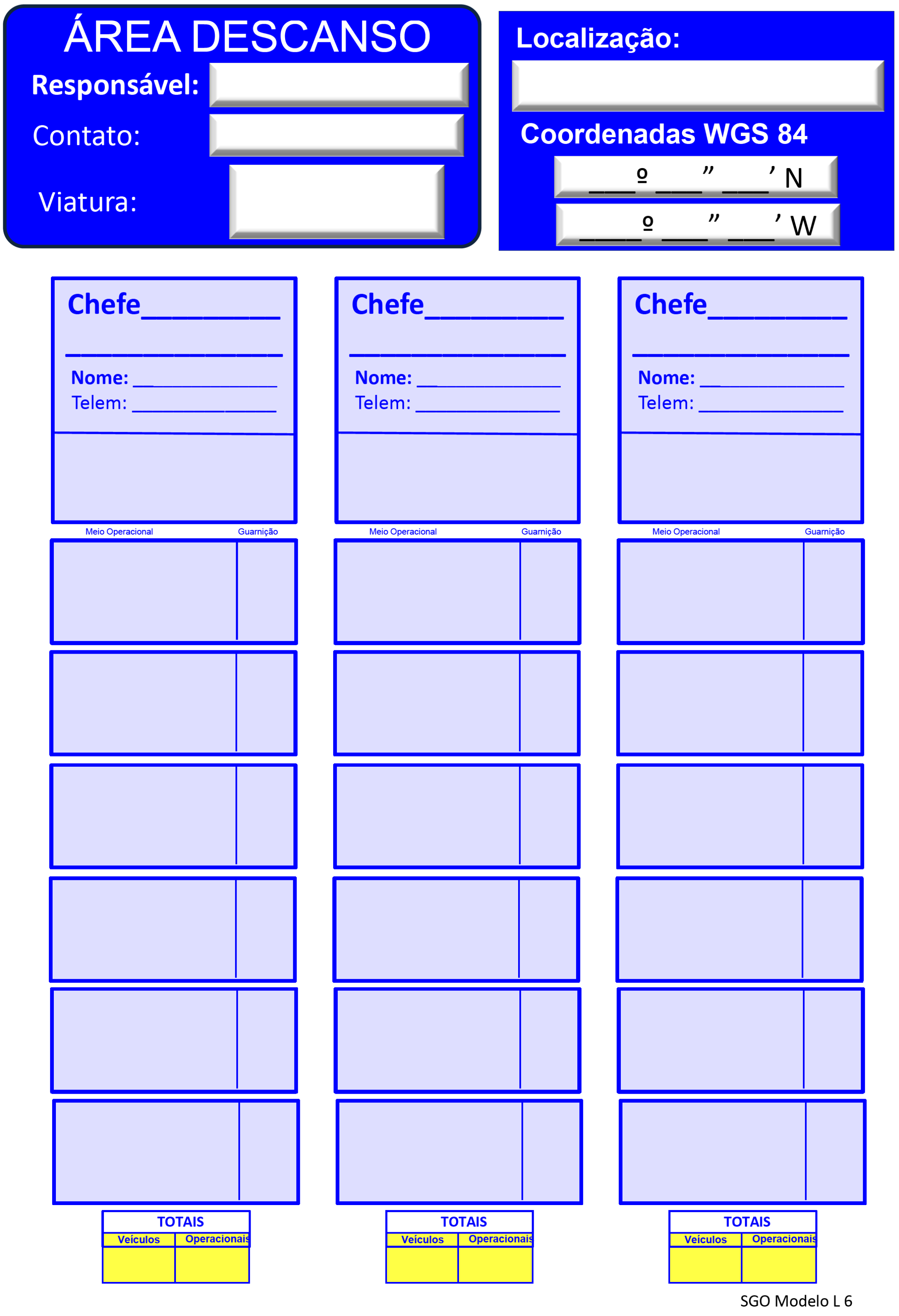
QUADRO DE MEIOS E LOGISTICA

SGO - FASE II

DATA INÍCIO ______ HORA : OCORRÊNCIA Nº ______ LOCAL / MUNICÍPIO ______

	PEDIDO	PREVISÃO DE CHEGADA	MEIO	ENTIDADE	Nº ELEMENTOS	HORA CHEGADA	EMPENHAMENTO								REFEIÇÃO		COMBUSTÍVEL		DESMOBILIZADO
							Hora	Destino	Hora	Destino	Hora	Destino	Hora	Destino			Hora	Litros	
1	:	:				:	:		:		:		:		:	:	:		
2	:	:				:	:		:		:		:		:	:	:		
3	:	:				:	:		:		:		:		:	:	:		
4	:	:				:	:		:		:		:		:	:	:		
5	:	:				:	:		:		:		:		:	:	:		
6	:	:				:	:		:		:		:		:	:	:		
7	:	:				:	:		:		:		:		:	:	:		
8	:	:				:	:		:		:		:		:	:	:		
9	:	:				:	:		:		:		:		:	:	:		
10	:	:				:	:		:		:		:		:	:	:		
11	:	:				:	:		:		:		:		:	:	:		
12	:	:				:	:		:		:		:		:	:	:		
13	:	:				:	:		:		:		:		:	:	:		
14	:	:				:	:		:		:		:		:	:	:		
15	:	:				:	:		:		:		:		:	:	:		
16	:	:				:	:		:		:		:		:	:	:		
17	:	:				:	:		:		:		:		:	:	:		
18	:	:				:	:		:		:		:		:	:	:		
19	:	:				:	:		:		:		:		:	:	:		
20	:	:				:	:		:		:		:		:	:	:		
21	:	:				:	:		:		:		:		:	:	:		
22	:	:				:	:		:		:		:		:	:	:		
23	:	:				:	:		:		:		:		:	:	:		
24	:	:				:	:		:		:		:		:	:	:		
25	:	:				:	:		:		:		:		:	:	:		
26	:	:				:	:		:		:		:		:	:	:		
27	:	:				:	:		:		:		:		:	:	:		

FIGURE D.13: Logistics and resources form.



ÁREA DESCANSO

Responsável:

Contato:

Viatura:

Localização:

Coordenadas WGS 84

___º ___” ___’ N

___º ___” ___’ W

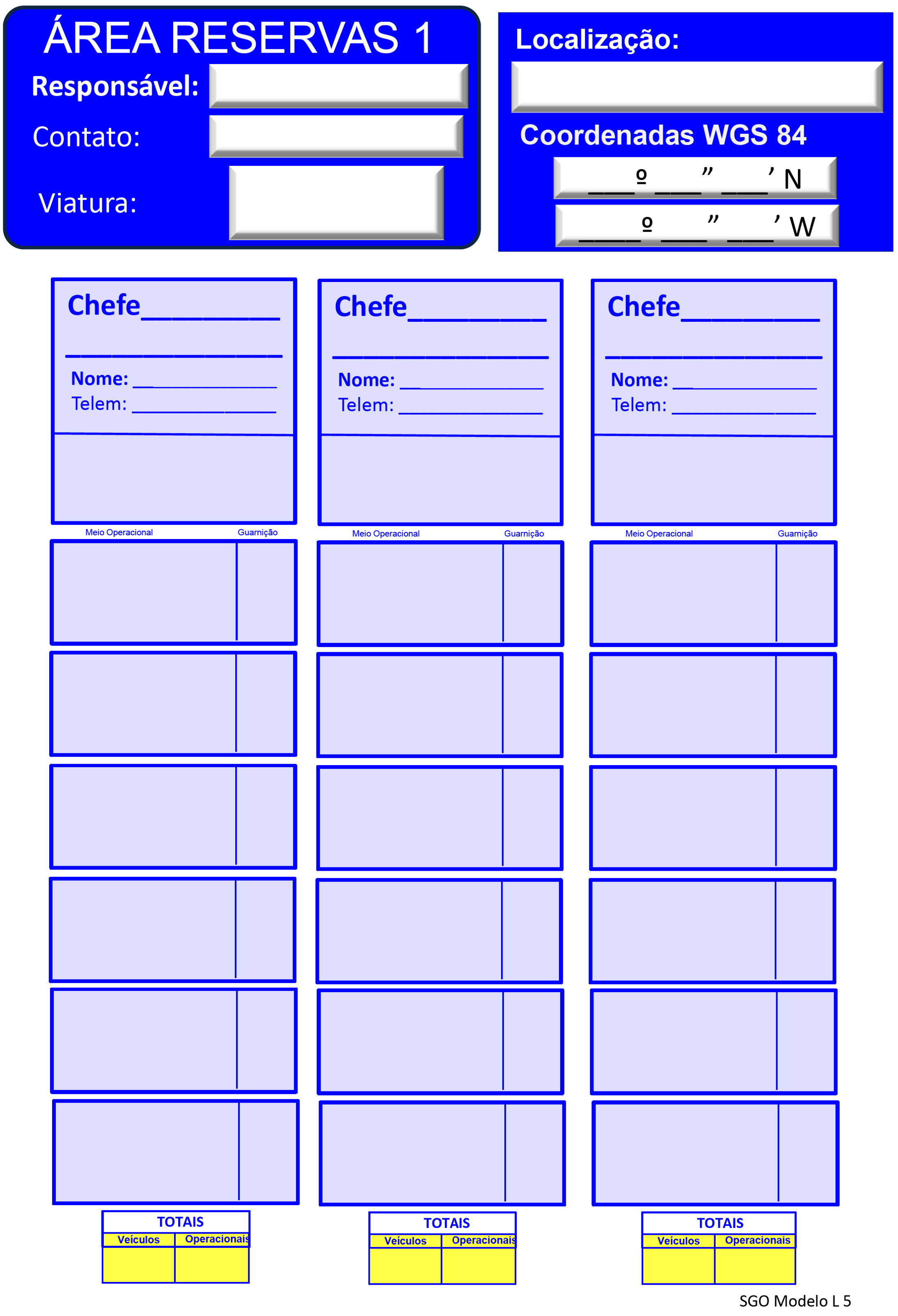
Chefe________	**Chefe**________	**Chefe**________
Nome: ________	**Nome:** ________	**Nome:** ________
Telem: ________	Telem: ________	Telem: ________

Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição

TOTAIS		TOTAIS		TOTAIS	
Veículos	Operacionais	Veículos	Operacionais	Veículos	Operacionais

SGO Modelo L 6

Figure D.14: Reserve and concentration form.



ÁREA RESERVAS 1

Responsável:

Contato:

Viatura:

Localização:

Coordenadas WGS 84

___º ___” ___’ N

___º ___” ___’ W

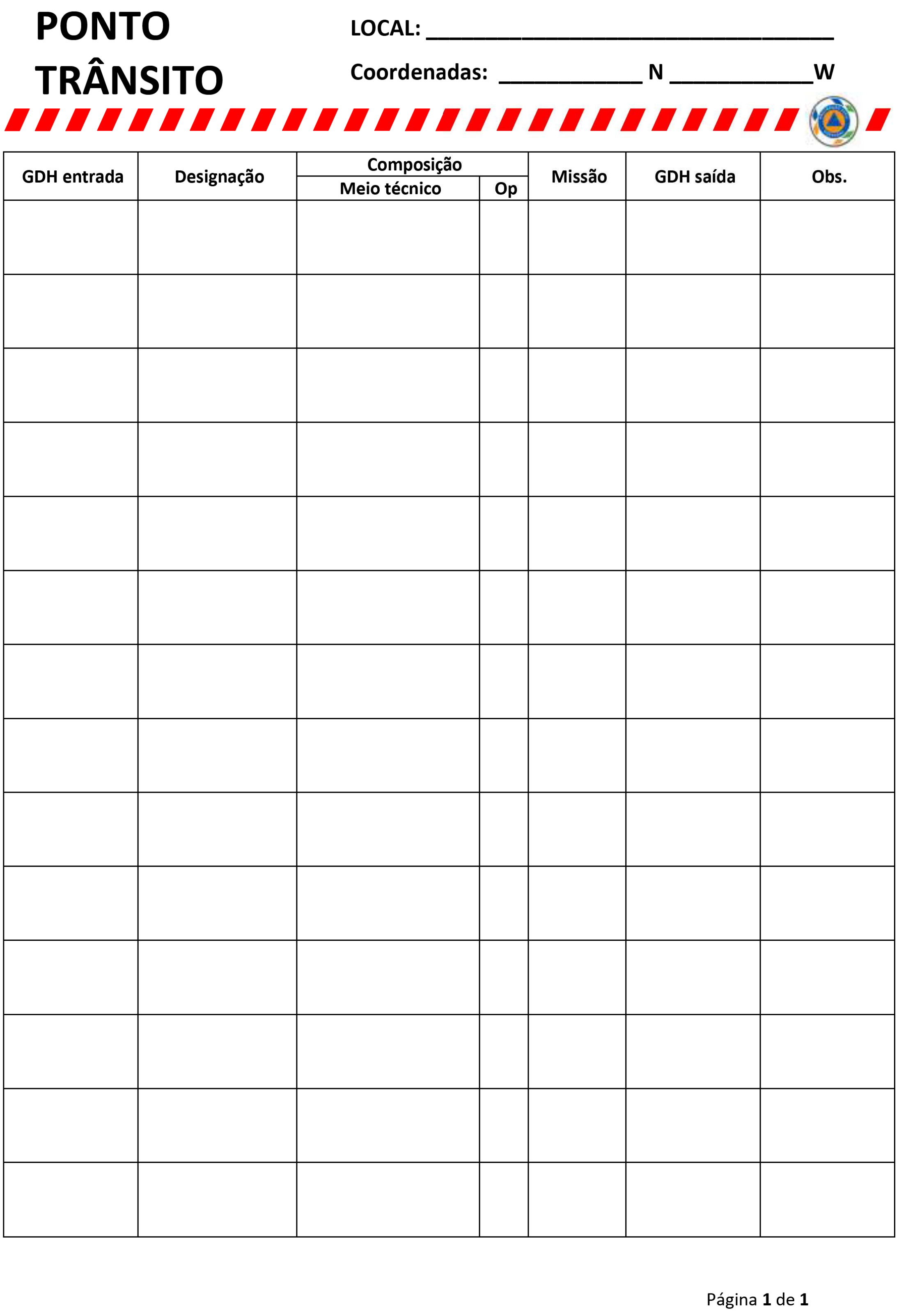
Chefe__________	Chefe__________	Chefe__________
Nome: ____________	Nome: ____________	Nome: ____________
Telem: ____________	Telem: ____________	Telem: ____________

Meio Operacional	Guarnição	Meio Operacional	Guarnição	Meio Operacional	Guarnição

TOTAIS		TOTAIS		TOTAIS	
Veículos	Operacionais	Veículos	Operacionais	Veículos	Operacionais

SGO Modelo L 5

Figure D.15: Reserve area form.



PONTO TRÂNSITO

LOCAL: ______________________________

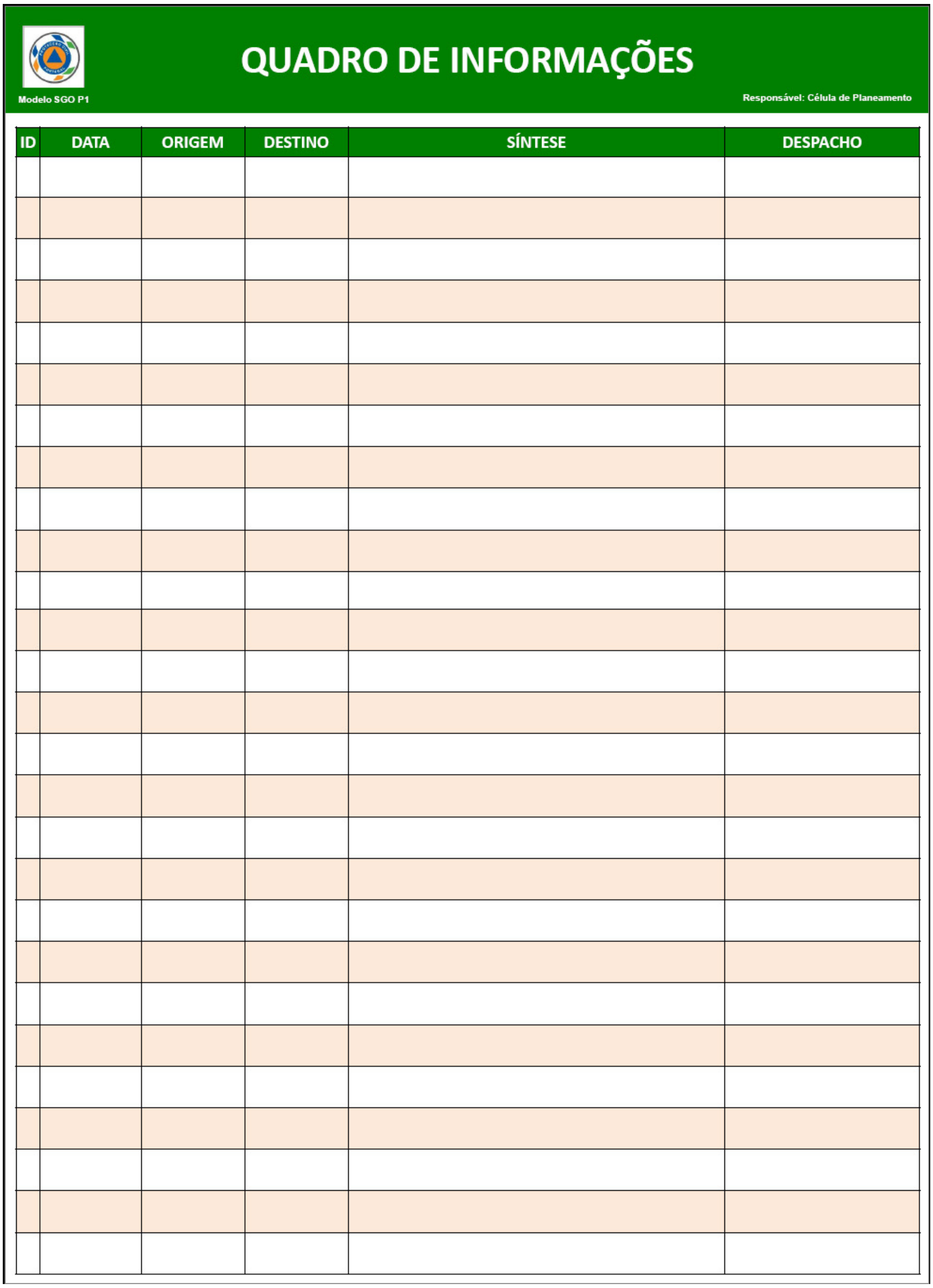
Coordenadas: ____________ **N** ____________**W**

GDH entrada	Designação	Composição		Missão	GDH saída	Obs.
		Meio técnico	Op			

Página **1** de **1**

Figure D.16: Traffic point form.

D.3 CEPLAN forms



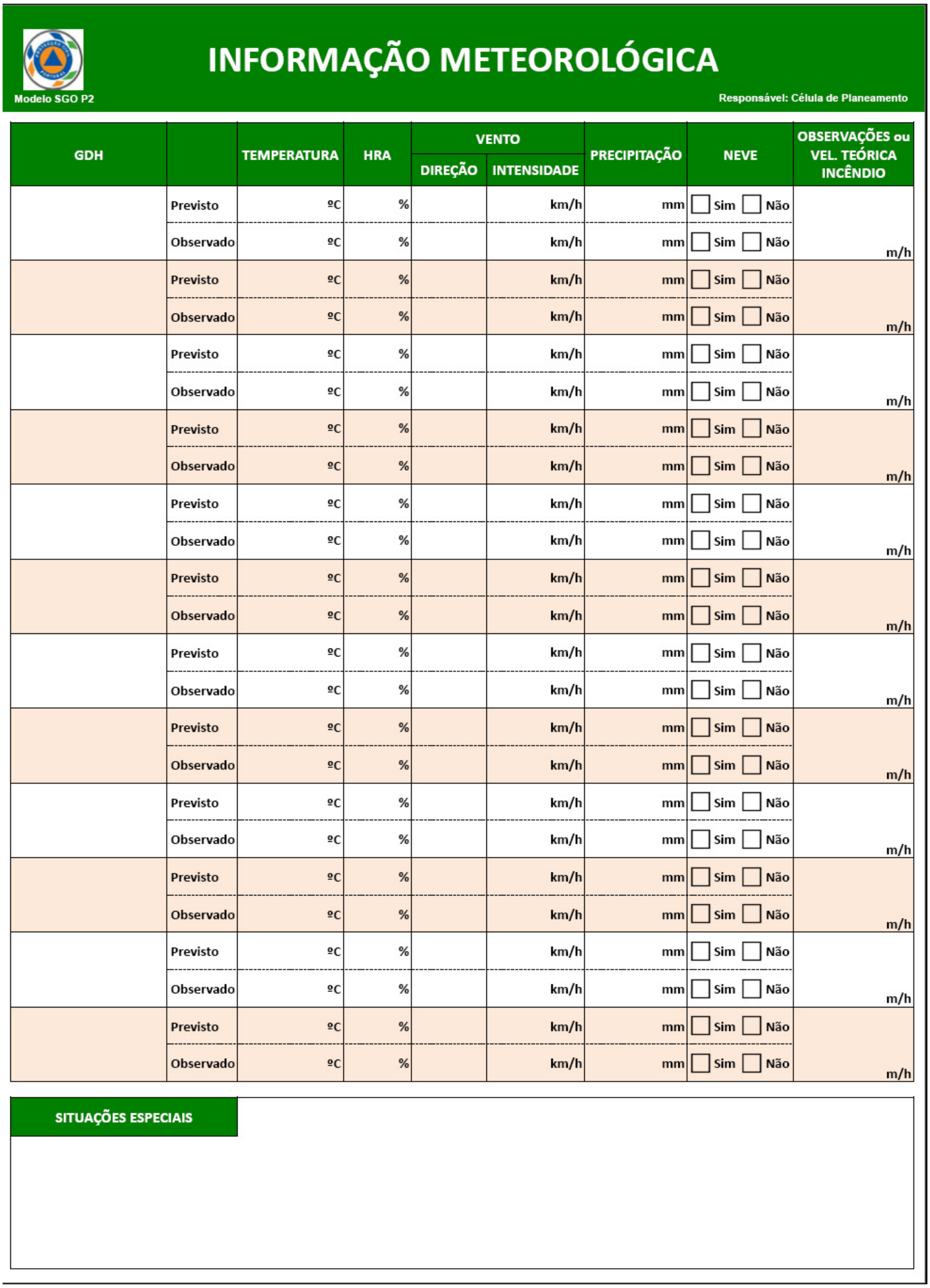
Modelo SGO P1

QUADRO DE INFORMAÇÕES

Responsável: Célula de Planeamento

ID	DATA	ORIGEM	DESTINO	SÍNTESE	DESPACHO

FIGURE D.17: Information form.



INFORMAÇÃO METEOROLÓGICA

Modelo SGO P2

Responsável: Célula de Planeamento

GDH		TEMPERATURA	HRA	VENTO		PRECIPITAÇÃO	NEVE	OBSERVAÇÕES ou VEL. TEÓRICA INCÊNDIO
				DIREÇÃO	INTENSIDADE			
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h
	Previsto	ºC	%		km/h	mm	☐ Sim ☐ Não	
	Observado	ºC	%		km/h	mm	☐ Sim ☐ Não	m/h

SITUAÇÕES ESPECIAIS

FIGURE D.18: Meteorologic Information form.

Bibliography

[ADB^{+}99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[Ash07] Mark Ashdown. Asymmetric distributed collaboration in emergency response. Technical report, Marie Curie Outgoing International Fellowship, 2007.

[Bre05] B Brehmer. The dynamic ooda loop: Amalgamating boyds ooda loop and the cybernetic approach to command and control. In *15 th International Command and Control Research and Technology Symposium (ICCRTS)*, 2005.

[Cro06] Douglas Crockford. The fat-free alternative to xml, 2006.

[CS99] MacKinley J. D. Card, S. K. and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[Des08] Despacho 20915/2008. Regulamento do modelo organizativo dos corpos de bombeiros. D.R. n.º 154, Série II de 2008-08-11, Ministério da Administração Interna, Autoridade Nacional de Protecção Civil, 2008.

[Deu96] P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational), May 1996.

[DHYS04] Jill L. Drury, Dan Hestand, Holly A. Yanco, and Jean Scholtz. Design guidelines for improved human-robot interaction. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *CHI Extended Abstracts*, page 1540. ACM, 2004.

[EBJ03] M.R. Endsley, B. Bolte, and D.G. Jones. *Designing for Situation Awareness: An Approach to User-Centered Design*. Taylor & Francis, 2003.

[EGJ93] R.A. Earnshaw, M.A. Gigante, and H. Jones. *Virtual reality systems*. Academic Press, 1993.

[EGR91] C. A. Ellis, S. J. Gibbs, and G.L. Rein. Groupware: Some issues and experiences. In *Communications of the ACM*, pages 39–58, 1991.

[Fer13] Nuno Ferreira. Mrsensing - environmental monitoring and context recognition with cooperative mobile robots in catastrophic incidents. Master's thesis, Faculty of Sciences and Technology of University of Coimbra, Portugal, September 2013.

[Fit54] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

[GHC09] Patrick Grommes, Rob Hutton, and Nick Colford. Comprehensive model of first responder operations & concept of operations. Technical report, COPE, 2009.

[Gue05] António Matos Guerra. *Segurança e protecção individual.* Escola Nacional de Bombeiros, Sintra, 2. edição, revista e actualizada edition, 2005. Volume VIII.

[Hic52] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, March 1952.

[Huf52] David Huffman. A method for the construction of Minimum-Redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

[Hym53] Ray Hyman. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, 45:188–196, 1953.

[Lee13] David A. Lee. Fat markup: Trimming the myth one calorie at a time. In *In Proceedings of Balisage: The Markup Conference 2013*, 2013.

[LW09] Michael Lewis and Huadong Wang. Using humans as sensors in robotic search, 2009.

[Mac92] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7(1):91–139, March 1992.

[Mar13] Joao Martins. MRSLAM - Multi-Robot Simultaneous Localization and Mapping. Master's thesis, Faculty of Sciences and Technology of University of Coimbra, Portugal, September 2013.

[Mil56] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. 62:81–97+, 1956.

[MP07] Christopher A. Miller and Raja Parasuraman. Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control. *Human Factors*, 49:57–75, 2007.

[MWH04] Bruce A. Maxwell, Nicolas Ward, and Frederick Heckel. Game-based design of human-robot interfaces for urban search and rescue. In *In Computer-Human Interface Fringe*, 2004.

[NOP09a] Norma Operacional Permanente. Classificação de Ocorrências. NOP 3101/09, Ministério da Administração Interna, Comando Nacional de Operações de Socorro, 2009.

[NOP09b] Norma Operacional Permanente. Sistema de Gestão de Operações - Simbologia. NOP 1402/09, Ministério da Administração Interna, Comando Nacional de Operações de Socorro, 2009.

[NOP12] Norma Operacional Permanente. Sistema de Gestão de Operações - SGO. NOP 1401/12, Ministério da Administração Interna, Comando Nacional de Operações de Socorro, 2012.

[Oli12] Vinicius Oliveira. Tufao. https://github.com/vinipsmaker/tufao, 2012.

[Pas98] Mr. Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Proceedings of the 2Nd IEEE International Symposium on Wearable Computers*, ISWC '98, pages 92–, Washington, DC, USA, 1998. IEEE Computer Society.

[Pin13] Andrea T. Pinto. iBombeiro. Master's thesis, Faculty of Sciences and Technology of University of Coimbra, Portugal, September 2013.

[Ras93] J. Rasmussen. Deciding and doing: Decision making in natural contexts. In G.A. Klein, J. Orasanu, and R. Calderwood, editors, *Decision Making in Action: Models and Methods*, pages 158–171. Ablex Publishing Corporation, 1993.

[RSP11] Y. Rogers, H. Sharp, and J. Preece. *Interaction Design: Beyond Human - Computer Interaction*. Interaction Design: Beyond Human-computer Interaction. Wiley, 2011.

[Saf07] D. Saffer. *Designing for Interaction: Creating Smart Applications And Clever Devices*. Voices that matter. New Riders Publishing, 2007.

[SAW94] Bill N. Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

[Sha48] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July, October 1948.

[Sin84] R. W. Sinnott. Virtues of the Haversine. , 68:158, December 1984.

[SOG06] Standard Operating Guideline. Extreme fire behavior. SOG 242/2006/10, Fire2000, 3D Firefighting, 2006.

[SOP09] Standard Operating Procedure. Tactical deployment into fire involved structures. SOP 1/V2/09, Paul Grimwood, Firetactics.com, 2009.

[SR96] Mike Scaife and Yvonne Rogers. External cognition: How do graphical representations work? *Int. J. Hum.-Comput. Stud.*, 45(2):185–213, August 1996.

[SSB04] E. Salas, K. C. Stagl, and C. S. Burke. 25 years of team effectiveness in organizations: research themes and emerging needs. *International Review of Industrial and Organizational Psychology, John Wiley & Sons, Chichester*, 19, 2004.

[YD04] Holly A. Yanco and Jill L. Drury. Classifying human-robot interaction: an updated taxonomy. In *SMC (3)*, pages 2841–2846. IEEE, 2004.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.