Crisis Management Systems
A Case Study for Aspect-Oriented Modeling

Jörg Kienzle[1], Nicolas Guelfi[2] and Sadaf Mustafiz[1]

with contributions from

Christian Fischer, Damien Garot, Laurent Vuillermoz, Jacques Klein, Alfredo Capozucca, Florencia Balbastro

[1]School of Computer Science, McGill University, Montreal, Canada

[2]Laboratory of Advanced Software Systems, University of Luxembourg, Luxembourg

Nicolas.Guelfi@uni.lu, Joerg.Kienzle@mcgill.ca, sadaf@cs.mcgill.ca

SOCS-TR-2009.3

Version 1.0.1

Abstract

The intent of this document is to define a common case study for the aspect-oriented modeling research community. The domain of the case study is crisis management systems, i.e., systems that help in identifying, assessing, and handling a crisis situation by orchestrating the communication between all parties involved in handling the crisis, by allocating and managing resources, and by providing access to relevant crisis-related information to authorized users. This document contains informal requirements of crisis management systems (CMS) in general, a feature model for a CMS product line, use case models for a car crash CMS (CCCMS), a domain model for the CCCMS, an informal physical architecture description of the CCCMS, as well as some design models of a possible object-oriented implementation of parts of the CCCMS backend. AOM researchers that want to demonstrate the power of their AOM approach or technique can hence apply the approach at the most adequate level of abstraction.

Change Log

Date	Version	Change
12/4/2009	1.0	Initial release
26/5/2009	1.0.1	Corrected some minor errors in use case 4.2.6, figure 6 and figure 8.

Motivation

Aspect-oriented modeling (AOM) has grown out of its infancy. There now exist many AOM techniques for different modeling notations (class diagrams, sequence diagrams, state diagrams, protocol machines, component diagrams, live sequence charts, use cases, etc..) and many AOM approaches that can be applied at different phases of software development (requirements, analysis, architecture, design, implementation) or to different domains (software product lines, security, fault tolerance, etc...). A considerable number of papers in journals and high-quality conferences have been published on AOM, and 13 successful workshops on AOM have been organized at the AOSD conferences and MoDELS conferences in the last 7 years. There is also the series of workshops entitled *Early Aspects* that falls within the context of AOM. Finally, a special issue of Transactions on Aspect-Oriented Software Development on *Aspects and Model-Driven Engineering* is currently in the process of being published.

Despite the many sources of information on AOM, it is not easy for people who want to use AOM to choose an appropriate AOM technique. Comparing different AOM approaches with each other to highlight the advantages and disadvantages of each one is not trivial, because each research team applies their AOM approach within a different context. Finally, it is not clear nowadays how several AOM approaches that apply to different phases of software development can be combined to produce a coherent *aspect-oriented software development process*.

This is why we propose a common case study to the AOM research community. The case study is chosen in such a way that all AOM approaches should be applicable to (some parts of) it. We will gather the results (papers / technical reports / models) of all research groups that decide to use this case study to demonstrate the power of their AOM approach or techniques. The resulting collection of information will provide a valuable resource for many researchers:

- Researchers that want to learn about AOM will find a concise collection of descriptions of solid and mature AOM approaches. They will only have to understand one case study in order to appreciate the sample models shown in every paper.
- Researchers that want to apply AOM for a particular purpose and are looking for the most appropriate AOM technique will be able to identify the most promising approach(es) easily. Identifying similarities between their problem and the case study will help them to determine candidate approaches.
- AOM experts can readily identify approaches that were able to handle concerns that their own approach can not handle elegantly. This stimulates cross-fertilization between approaches and collaborative research, and brings the AOM community closer to the definition of an aspect-oriented software development process that covers all software development phases.

The remainder of this document describes the case study in detail. The domain of the case study is *crisis management*. In order to make sure that all AOM approaches and techniques are somehow applicable to this case study, it is on purpose defined in a very broad way. The description of the case study includes non-formal requirements, use cases / activity diagrams, a domain model, an informal architecture description, and some detailed design models using standard, non-aspect-oriented notations.

AOM approaches that apply to early phases of software development should either work with the general crisis management system requirements, or use the more specific car crash crisis management system requirements. If the AOM approach you want to present applies to a late software development phase such as design, we request that you apply your approach to the car crash crisis management system backend / server.

Contents

List of Figures

1 Crisis Management System Case Study Overview

The domain of the case study is *crisis management systems* (CMS). The need for crisis management systems has grown significantly over time. A crisis can range from major to catastrophic affecting many segments of society. Natural disasters (e.g. earthquakes, tsunamis, twisters, fire, floods, etc...), terrorist attacks or sabotage (explosions, kidnapping, etc...), accidents (plant explosion, pollution emergency, a car crash, etc...), and technological disruptions are all examples of emergency situations that are unpredictable and can lead to severe after-effects unless handled immediately. Crisis management involves identifying, assessing, and handling the crisis situation. A crisis management system facilitates this process by orchestrating the communication between all parties involved in handling the crisis. The CMS allocates and manages resources, and provides access to relevant crisis-related information to authorized users of the CMS.

Different existing AOM approaches and techniques are meant to be used during different phases of software development. As a result, different AOM approaches work with different kinds of models and modeling notations at different levels of abstraction. In order to make sure that all AOM approaches and techniques are somehow applicable to this case study, we present a set of models in this technical report:

1. Short, *informal requirements* text describing the domain of crisis management systems in more detail. It also mentions some non-functional requirements of a CMS, e.g. security and dependability. This text probably contains information that is important to everyone that wants to work on this case study. The information description is presented in Section 2 on page 7.

2. *Feature diagrams* highlighting the software product line aspect of crisis management systems. Crisis management systems can be used to handle many types of crises (e.g., natural disasters, epidemics, accidents, attacks, etc...) and may have to interface and interoperate with different types of external services (e.g., military systems, police systems, government, medical services, etc...). The feature diagram models are presented in Section 3 on page 12.

3. *Use cases* describing a particular CMS suitable for dealing with *car crash crises*. The Car Crash CMS use case model description can be found in Section 4 on page 15.

4. A domain model of the car crash crisis management system that documents the key concepts, and the domain-vocabulary of the Car Crash CMS is presented in Section 5 on page 22.

5. An *activity diagram* that shows how the extended use case model can be translated into a formal requirements specification. An example activity diagram model is shown in Section 6 on page 24.

6. An informal description of a possible physical architecture for the car crash crisis management system is presented in Section 7 on page 26.

7. Some detailed design models for the car crash crisis management system backend are given in Section 8 on page 27.

AOM approaches that apply to early phases of software development should either work with the general crisis management system requirements, or use the more specific car crash crisis management system requirements. If the AOM approach you want to present applies to a late software development phase such as design, we request that you apply your approach to the car crash crisis management system backend / server.

You are not allowed to add new functional or non-functional requirements to the case study. On the other hand, you are allowed to correct eventual errors or clarify ambiguities in the document and the models provided you justify the need to do so.

2 Crisis Management System: Requirements

The user requirements outlined in this section are based on [4]. The general objectives of a crisis management system (CMS) include the following:

- To help in the coordination and handling of a crisis;
- To ensure that an abnormal or catastrophic situation does not go out of hand;
- To minimize the crisis by handling the situation using limited resources;
- To allocate and manage resources in an effective manner;
- To identify, create, and execute missions in order to manage the crisis;
- To archive the crisis information to allow future analysis.

2.1 Crisis Scenario (of a Car Crash Crisis Management System)

A crisis management scenario is usually triggered by a crisis report from a witness at the scene. A coordinator, who is in charge of organizing all required resources and tasks, initiates the crisis management process. The coordinator has access to the camera surveillance system. The surveillance system is an external system used to monitor traffic on highways or other busy routes. The cameras are installed only in specific locations. If a crisis occurs in locations under surveillance, the crisis management system can request video feed that allows the coordinator to verify the witness information.

A super observer, an expert in the field (depending on the kind of crisis), is assigned to the scene to observe the emergency situation and identify the tasks necessary to cope with the situation. The tasks are crisis missions defined by the observer. The coordinator is then required to process the missions by allocating suitable resources to each task.

Depending on the type of crisis, human resources could include firemen, doctors, nurses, policemen, and technicians, and hardware resources could include transportation systems, computing resources, communication means (such as PDAs or mobile phones), or other necessities like food or clothes. Animals, for instance police dogs, are also used as resources in some situations. The human and animal resources act as first-aid workers. Each first-aid worker is assigned a specific task which needs to be executed to recover from the abnormal situation. The workers are expected to report on the success or failure in carrying out the missions. The completion of all missions would allow the crisis to be concluded.

2.2 Scope of the CMS

A crisis management system (CMS) should include the following functionalities:

- initiating a crisis based on an external input from a witness,
- processing a crisis by executing the missions defined by a super observer and then assigning internal and/or external resources,
- wrapping-up and archiving crisis,
- authenticating users,
- handling communication between coordinator/system and resources.

CMS replace existing crisis management systems that a) still manually keep track of important crisis-related information and that b) operate largely without automated support for crisis resolution strategies in order to respond to a crisis.

2.3 Non-functional Requirements of the CMS

The crisis management system shall exhibit the following non-functional properties:

- **Availability**
  - The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
  - The system shall recover in a maximum of 30 seconds upon failure.
  - Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.
- **Reliability**
  - The system shall not exceed a maximum failure rate of 0.001%.
  - The mobile units shall be able to communicate with other units on the crisis site and the control centre regardless of location, terrain and weather conditions.
- **Persistence**
  - The system shall provide support for storing, updating and accessing the following information on both resolved and on-going crises: type of crisis; location of crisis; witness report; witness location; witness data; time reported; duration of resolution; resources deployed; civilian casualties; crisis management personnel casualties; strategies used; missions used; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; log of communications; log of decisions; log of problems encountered.
  - The system shall provide support for storing, updating and accessing the following information on available and deployed resources (both internal and external): type of resource (human or equipment); capability; rescue team; location; estimated time of arrival (ETA) on crisis site.
  - The system shall provide support for storing, updating and accessing the following information on crisis resolution strategies: type of crisis; step-by-step guide to resolve crisis; configuration of missions required; links to alternate strategies; applications to previous crises; success rate.
- **Real-time**
  - The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.

  - The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
  - The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.

- **Security**
  - The system shall define access policies for various classes of users. The access policy shall describe the components and information each class may add, access and update.
  - The system shall authenticate users on the basis of the access policies when they first access any components or information. If a user remains idle for 30 minutes or longer, the system shall require them to re-authenticate.
  - All communications in the system shall use secure channels compliant with AES-128 standard encryption.

- **Mobility**
  - Rescue resources shall be able to access information on the move.
  - The system shall provide location-sensitive information to rescue resources.
  - Rescue resources shall communicate their location to the control centre.
  - The system shall have access to detailed maps, terrain data and weather conditions for the crisis location and the routes leading to it.

- **Statistic Logging**
  - The system shall record the following statistical information on both on-going and resolved crises: rate of progression; average response time of rescue teams; individual response time of each rescue team; success rate of each rescue team; rate of casualties; success rate of missions.
  - The system shall provide statistical analysis tools to analyse individual crisis data and data on multiple crises.

- **Multi-Access**
  - The system shall support at least 1000 witnesses calling in at a time.
  - The system shall support communication, coordination and information access for at least 20000 rescue resources in deployment at a time.
  - The system shall support management of at least 100 crises at a time.
  - The system shall support management of at least 200 missions per crisis at a time.

- **Safety**
  - The system shall monitor emissions from crisis site to determine safe operating distances for rescue resources.
  - The system shall monitor weather and terrain conditions at crisis site to ensure safe operation and withdrawal of rescue resources, and removal of civilians and casualties.

  - The system shall determine a perimeter for the crisis site to ensure safety of civilians and removal of casualties to a safe distance.
  - The system shall monitor criminal activity to ensure safety of rescue resources, civilians and casualties.
  - The safety of rescue personnel shall take top priority for the system.

- **Adaptability**
  - The system shall recommend alternate strategies for dealing with a crisis as the crisis conditions (e.g., weather conditions, terrain conditions, civilian or criminal activity) change.
  - The system shall recommend or enlist alternate resources in case of unavailability or shortage of suitable resources.
  - The system shall be able to use alternate communication channels in case of unavailability or shortage of existing channels.
  - The system shall be able to maintain effective communication in areas of high disruption or noise at the crisis site.

- **Accuracy**
  - The system shall have access to map, terrain and weather data with a 99% accuracy.
  - The system shall provide up-to-date information to rescue resources.
  - The system shall record data upon receipt without modifications.
  - The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.

2.4 Car Crash Crisis Management System

In some of the models presented in this technical report, we have focused on one particular CMS: the car crash crisis management system. The car crash CMS includes all the functionalities of general crisis management systems, and some additional features specific to car crashes such as facilitating the rescuing of victims at the crisis scene and the use of tow trucks to remove damaged vehicles.

2.4.1 Scope of the Car Crash CMS

A car crash is defined in Wikipedia as following:

> A car accident or car crash is an incident in which an automobile collides with anything that causes damage to the automobile, including other automobiles, telephone poles, buildings or trees, or in which the driver loses control of the vehicle and damages it in some other way, such as driving into a ditch or rolling over. Sometimes a car accident may also refer to an automobile striking a human or animal.

Our Car Crash CMS addresses car crashes involving single or multiple vehicles, humans, or other objects. This case study is however limited to management of human victims only and does not provide rescue missions specifically for animals. First-aid animal workers are not included in the scope of this case study either.

Car crash specific functionalities include the following:

- facilitating the rescue mission carried out by the police by providing them with detailed information on the location of the crash;
- managing the dispatch of ambulances or other alternate emergency vehicles to transport victims from the crisis scene to hospitals;
- facilitating the first-aid missions by providing relevant medical history of identified victims to the first-aid workers by querying data bases of local hospitals;
- facilitating the medical treatment process of victims by providing important information about the crash to the concerned workers, i.e. paramedics, doctors, upon arrival at the hospital;
- managing the use of tow trucks to remove obstacles and damaged vehicles from the crisis scene.

2.4.2 Car Crash CMS Actors

The actors involved in the Car Crash CMS are defined in this section.

- **Coordinator** oversees management of the crisis by coordinating the resources and communicating with all the CMS employees and external workers.
- **Super Observer** is dispatched to the crisis scene to evaluate the situation and define the necessary missions to cope with the crisis.
- **CMS Employee** is an internal human resource who is qualified and capable of performing missions related to his field of expertise. The worker acts as a facilitator actor when he is in charge of or operating local resources (for example, tow trucks or ambulances).
- **External Worker** is an external resource who is specialized and capable of performing missions related to his field of expertise. The worker acts as a facilitator actor when he is in charge of or operating external resources (for example, police trucks or fire trucks).
- **System Admin** is the specialist who maintains the system and creates all profiles of workers and resources to feed the crisis management database.
- **Witness** is the person who reports the crisis by calling the crisis management center.
- **Phone Company** is an external entity contacted for verification of witness purposes.
- **Surveillance System** is an external entity which monitors traffic in highways and cities with the use of cameras.

3 Feature Models

Since there are so many different kinds of crises, the domain of crisis management systems is very broad. However, any crisis management system has a common set of responsibilities and functionalities. It is therefore natural to build a framework or product line of crises management systems, which can be specialized to create crisis management systems for a particular kind of crisis and a particular context. A feature diagram listing many possible features of a crisis management system is given in figure 1. It has been taken from [3].

Selection of some features requires the selection of other features. Examples of such dependencies are:

- *Natural Disasters* requires *Fire Department* and *External Company*
- *Terrorist Attack* requires *Army Special Unit* and *Police* and *Police Special Unit* and *Public Hospital*
- *Major Accident* requires *Police* and *Fire Department* and *Public Hospital* and *Private Hospital* and *Independent First-Aid Doctor* and *Private Ambulance Company*
- *Plant Explosion* requires *Police* and *Fire Department* and *Public Hospital*
- *Nuclear Plant Explosion* requires *The Army* and *Army Special Unit*

Figure 2 presents a possible set of features selected for the car crash CMS.

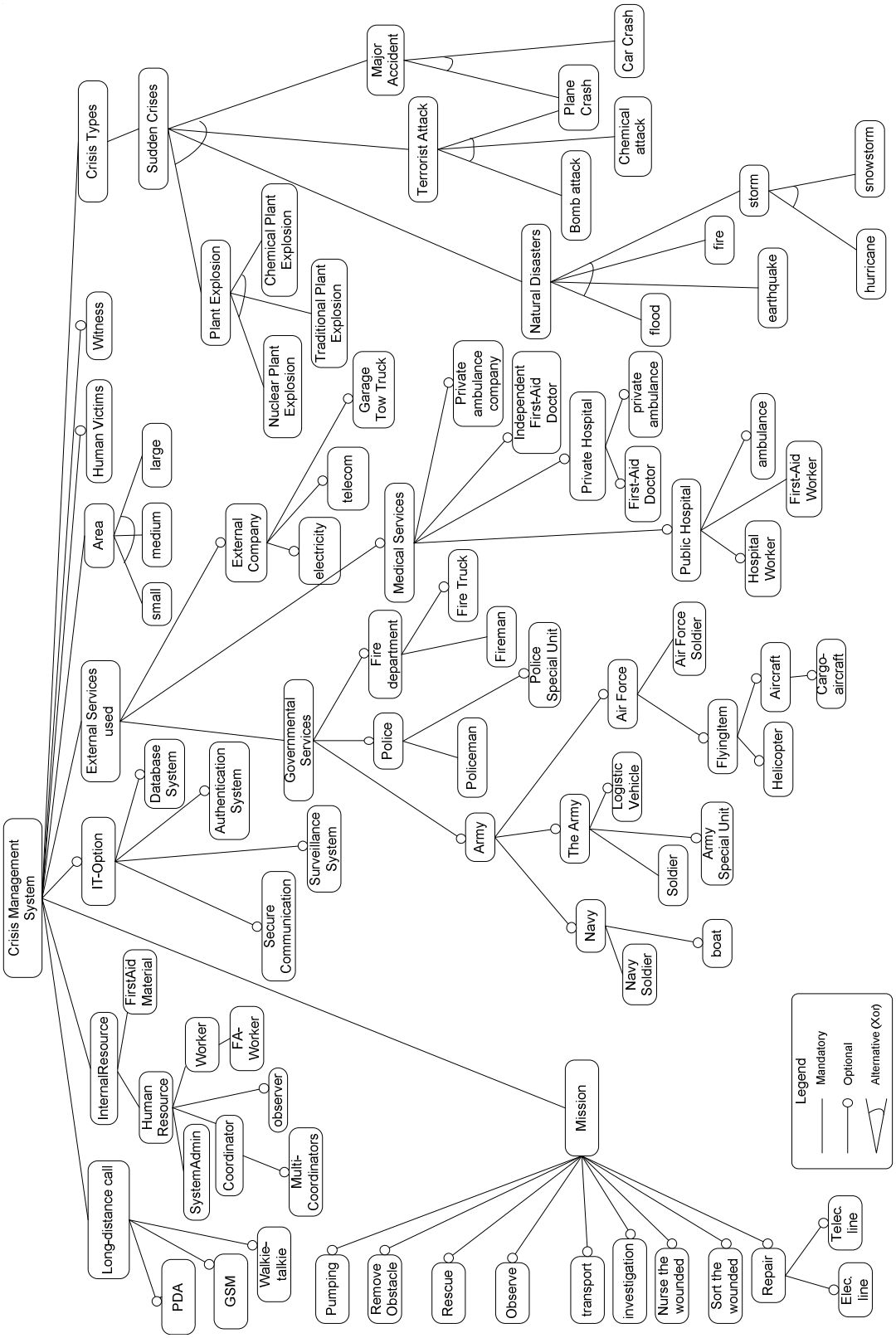

Figure 1: Crisis Management Systems Feature Diagram

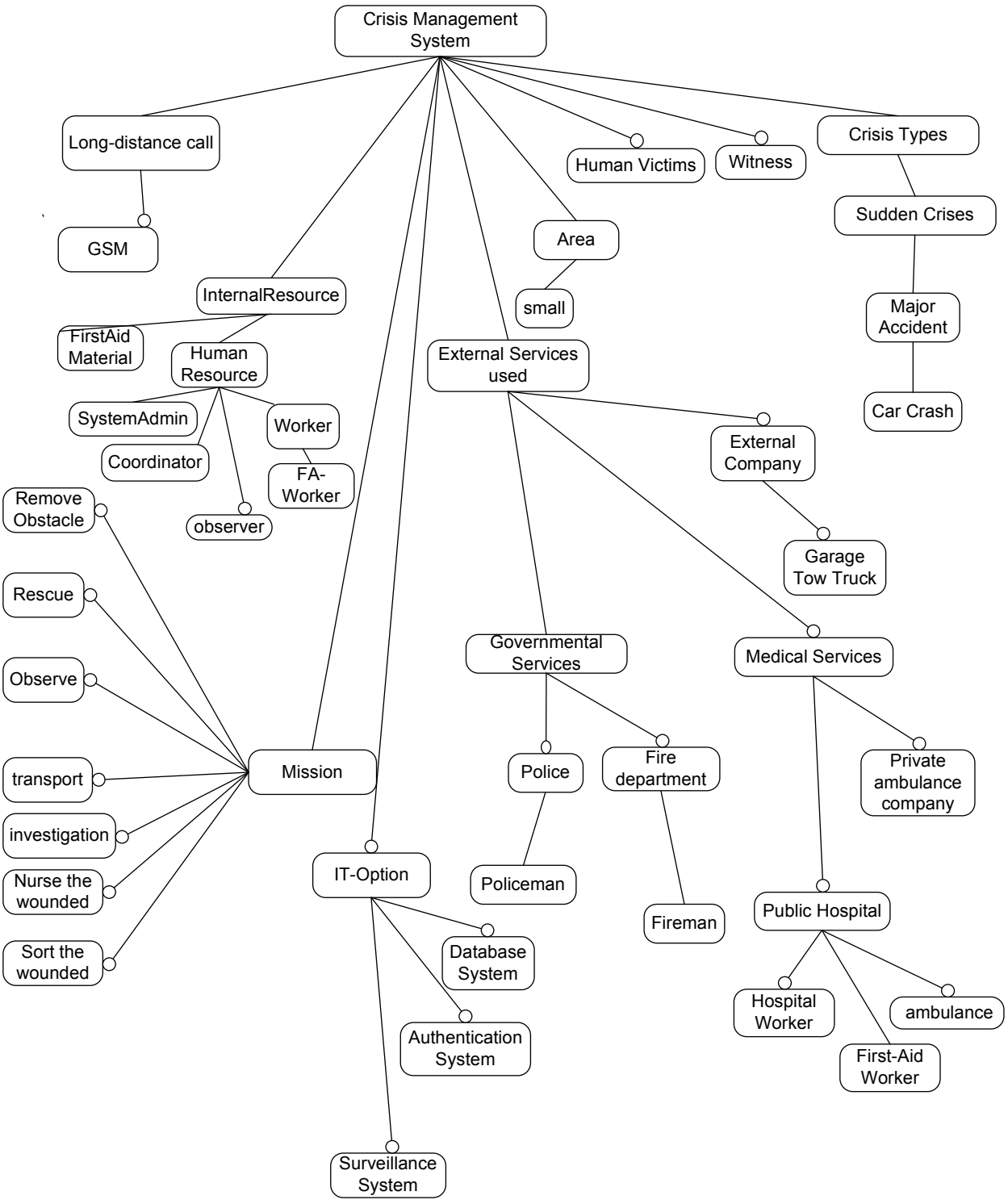

Figure 2: Car Crash Crisis Management Systems Feature Diagram

4 Use Cases

The use case model includes a summary use case diagram (presented in subsection 4.1, and individual use cases presented in subsection 4.2.

4.1 Use Case Diagram

Fig. 3 shows the use cases related to the summary-level goal *Resolve Crisis* in the Car Crash Crisis Management System, by means of a use case diagram.

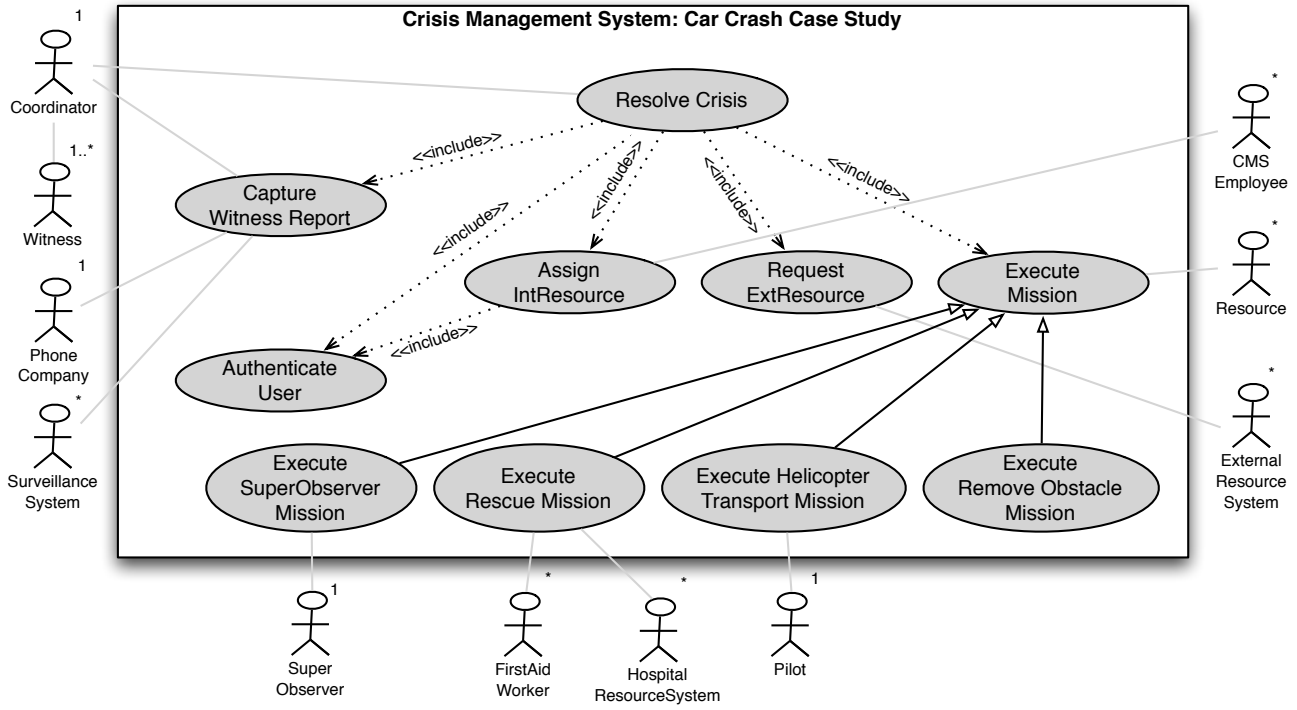

Figure 3: Car Crash Case: Standard Use Case Diagram

Details of all the use cases that directly relate to the summary level use case *Resolve Crisis* are given in section 4.2. The listed use cases are: Resolve Crisis, Capture Witness Report, Assign Internal Resource, Assign External Resource, Execute Mission, Execute SuperObserver Mission, Execute Rescue Mission, and Authenticate User.

Use cases describing other missions, such as the Execute Helicopter Transport Mission, or Execute Remove Obstacle Mission are not shown for space reasons. Likewise, details of use cases related to the management of the resource database are not included for space reasons. Such use cases would, for instance, include:

- Creating records for CMSEmployees
- Managing access rights of CMSEmployees
- Updating the availability of CMSEmployees due to sickness or vacation
- Dealing with problems of the CMS-controlled vehicles that are not related to a crisis

Finally, following a dependability-focussed requirements engineering process such as DREP [1], exceptional situations that a CMS might be exposed to should also be considered. For this case study, several exceptional situations were discovered that affect the context in which the system operates, and that require the system to react in a certain way to continue to provide reliable and safe service. The situations are:

- Severe Weather Conditions: Bad weather makes helicopter transportation impossible.
- Strike: A strike affects the availability of CMS employees and external workers.
- Risk of Explosion: Leaking gas and open fire threatens the safety of workers.
- VIP Victim: One of the crash victims is a VIP (such as for instance, the president). Handling of the crisis should therefore be coordinated by the appropriate office.
- Criminal Case: The reason for the crash is of criminal nature, and therefore the rescue missions have to be carried out accordingly.

To detect and to handle the above situations, we added the following exceptional actors: WeatherInformationSystem, NationalCrisisCenter. The detailed handler use cases that describe the functionality that such a reliable car crash CMS is to provide are not described in this document for space reasons.

4.2 Textual Use Cases

Use cases are a widely used formalism for discovering and recording behavioral requirements of software systems, since they can be effectively used as a communication means between technical as well as non-technical stakeholders of the software under development. In short, use cases are stories of using a system to meet goals. They are in general text-based, but their strength is that they both scale up or scale down in terms of sophistication and formality, depending on the need and context.

The use cases presented here follow a textual template. The *main success scenario* is a numbered list of lines of text (subsequently named *steps*) that describes the possible interactions between the primary actor, potential secondary actors and the Car Crash CMS (subsequently named *System*) that occur to reach a particular goal. Alternate ways of achieving a goal, or situations in which the goal can not be reached, are described in the *extension* part of the template.

4.2.1 Resolve Crisis

Use Case 1: Resolve Crisis
Scope: Car Crash Crisis Management System
Primary Actor: Coordinator
Secondary Actor: Resource
Intention: The intention of the Coordinator is to resolve a car crash crisis by asking employees and external workers to execute appropriate missions.
Main Success Scenario:

Witness places a call to the crisis centre, where it is answered by a Coordinator.

1. *Coordinator* captures witness report (UC 2).
2. *System* recommends to *Coordinator* the missions that are to be executed based on the current information about the crisis and resources.
3. *Coordinator* selects one or more missions recommended by the system.

For each mission in parallel:

4. For each internal resource required by a selected mission, *System* assigns an internal resource (UC 3).
5. For each external resource required by a selected mission, *System* requests an external resource (UC 4).
6. *Resource* notifies *System* of arrival at mission location.
7. *Resource* executes the mission (UC 5).
8. *Resource* notifies *System* of departure from mission location.
9. In parallel to steps 6-8, *Coordinator* receives updates on the mission status from *System*.
10. In parallel to steps 6-8, *System* informs *Resource* of relevant changes to mission / crisis information.
11. *Resource* submits the final mission report to *System*.

12. In parallel to steps 4-8, *Coordinator* receives new information about the crisis from *System*.
13. *Coordinator* closes the file for the crisis resolution.

Use case ends in success.

Extensions:

1a. *Coordinator* is not logged in.
- 1a.1 *Coordinator* authenticates with System (UC 10).
- 1a.2 Use case continues with step 1.

4a. Internal resource is not available after step 4.
- 4a.1 *System* requests an external resource instead (i.e., use case continues in parallel with step 5).

5a. External resource is not available after step 5.
- 5a.1 Use case continues in parallel with step 2.

6a. *System* determines that the crisis location is unreachable by standard transportation means, but reachable by helicopter.
- 6a.1 *System* informs the *Coordinator* about the problem.
- 6a.2 *Coordinator* instructs *System* to execute a helicopter transport mission (UC 09).
- 6a.3 Use case continues with step 6.

6b. *Resource* is unable to contact *System*.
- 6b.1 *SuperObserver* notifies *System* that resource arrived at the mission location.

6c. Although *Resource* should be at mission location by now, *Resource* has not yet notified *System*.
- 6c.1 *System* requests *Resource* to provide an update of its location.
- 6c.2 Use case continues at step 6.

7a. One or more further missions are required in step 6.
- 7a.1 Use case continues in parallel with step 2.

7b. The mission failed.
- 7b.1 Use case continues with step 2.

8a. *Resource* is unable to contact *System*.
- 8a.1 *SuperObserver* notifies *System* that resource is leaving the mission location.

8b. Although mission should be completed by now, *Resource* has not left mission location.
- 8b.1 *System* requests *Resource* to provide the reason for the delay.
- 8b.2 Use case continues at step 7.

9a. Changes to mission are required.
- 9a.1 Use case continues in parallel with step 2.

11a. *Resource* never files a mission report.
- 11a.1 Mission use case ends without mission report.

12a. Changes to mission are required.
- 12a.1 Use case continues in parallel with step 2.

4.2.2 Capture Witness Report

Use Case 2: Capture Witness Report
Scope: Car Crash Crisis Management System
Primary Actor: Coordinator
Secondary Actor: PhoneCompany, SurveillanceSystem
Intention: The Coordinator intends to create a crisis record based on the information obtained from witness.
Main Success Scenario:

Coordinator requests Witness to provide his identification.
1. *Coordinator* provides witness information[1] to *System* as reported by the witness.
2. *Coordinator* informs *System* of location and type of crisis as reported by the witness.

In parallel to steps 2-4:

2a.1 *System* contacts *PhoneCompany* to verify witness information.
2a.2 *PhoneCompany* sends address/phone information to *System*.
2a.3 *System* validates information received from the PhoneCompany.

3. *System* provides *Coordinator* with a crisis-focused checklist.
4. *Coordinator* provides crisis information[2] to *System* as reported by the witness.
5. *System* assigns an initial emergency level to the crisis and sets the crisis status to *active*.

Use case ends in success.

Extensions:

1a,2a. The call is disconnected. The base use case terminates.

In parallel to steps 3-4, if the crisis location is covered by camera surveillance:

3a.1 *System* requests video feed from *SurveillanceSystem*.
3a.2 *SurveillanceSystem* starts sending video feed to *System*.
3a.3 *System* starts displaying video feed for *Coordinator*.

4a. The call is disconnected.
4a.1 Use case continues at step 5 without crisis information.

5a. PhoneCompany information does not match information received from Witness.
5a.1 The base use case is terminated.

5b. Camera vision of the location is perfect, but *Coordinator* cannot confirm the situation that the witness describes *or* the *Coordinator* determines that the witness is calling in a fake crisis.
5b.1 The base use case is terminated.

[1]Witness information includes the first name, last name, phone number, and address.
[2]Crisis information includes the details about the crisis, the time witnessed, etc.

4.2.3 Assign Internal Resource

Use Case 3: Assign Internal Resource
Scope: Car Crash Crisis Management System
Primary Actor: None
Secondary Actor: CMSEmployee
Intention: The intention of System is to find, contact, and assign a mission to the most appropriate available CMSEmployee.
Main Success Scenario:

System selects an appropriate CMSEmployee based on the mission type, the emergency level, location and requested expertise. In very urgent cases, steps 1 and 2 can be performed for several CMSEmployees concurrently, until one of the contacted employees accepts the mission.
1. *System* sends *CMSEmployee* mission information.
2. *CMSEmployee* informs *System* that he accepts the mission.

Use case ends in success.

Extensions:

1a. *CMSEmployee* is not logged in.
1a.1 *System* requests the *CMSEmployee* to login.
1a.2 *CMSEmployee* authenticates with System (UC 10).
1a.3 Use case continues at step 1.
1b. *CMSEmployee* is unavailable or unresponsive.
1b.1 *System* selects the next appropriate *CMSEmployee*.
1b.2 Use case continues at step 1.
1b.1a No other *CMSEmployee* is available. Use case ends in failure.
2a. *CMSEmployee* informs *System* that he cannot accept the mission.
2a.1 *System* selects the next appropriate *CMSEmployee*.
2a.2 Use case continues at step 1.
2a.2a No other *CMSEmployee* is available. Use case ends in failure.

4.2.4 Request External Resource

Use Case 4: Request External Resource
Scope: Car Crash Crisis Management System
Primary Actor: Coordinator
Secondary Actor: ExternalResourceSystem (ERS)
Intention: The System requests a mission from an external resource, such as a fire station, police station or external ambulance service.
Main Success Scenario:

1. *System* sends mission request to *ERS*, along with mission-specific information[1].
2. *ERS* informs *System* that request can be processed.

Use case ends in success.

Extensions:

2a. *ERS* notifies *System* that it partially approves request for resources. Use case ends in degraded success.
2b. *ERS* notifies *System* that it can not service the request. Use case ends in failure.

[1]Mission-specific information includes things such as the location and emergency level of the mission, the quantity of vehicles requested, special characteristics of the aid worker or vehicle, etc.

4.2.5 Execute Mission

Use Case 5: Execute Mission
Intention: The Resource executes a mission in order to help resolve a crisis. ExecuteMission is an abstract use case. The details of the interaction for specific missions are presented in child use cases such as ExecuteSuperObserverMission (UC 6), or ExecuteRescueMission (UC 7).

4.2.6 Execute SuperObserver Mission

Use Case 6: Execute SuperObserver Mission
Scope: Car Crash Crisis Management System
Primary Actor: SuperObserver
Secondary Actor: None
Intention: The intention of the SuperObserver is to observe the situation at the crisis site to be able to order appropriate missions.

Main Success Scenario:

SuperObserver is at the crisis location.

1. *System* sends a crisis-specific checklist to *SuperObserver*.
2. *SuperObserver* feeds *System* with crisis information.
3. *System* suggests crisis-specific missions to *SuperObserver*.

Steps 4-8 is repeated as many times as needed.

4. *SuperObserver* notifies *System* of the type of mission he wants to create.
5. *System* sends a mission-specific information request to *SuperObserver*.
6. *SuperObserver* sends mission-specific information[1] to *System*.
7. *System* acknowledges the mission creation to *SuperObserver*.
8. *System* informs *SuperObserver* that mission was completed successfully.
9. *SuperObserver judges that his presence is no longer needed at the crisis location.*

Use case ends in success.

Extensions:

7a. Mission cannot be created and replacement missions are possible.
  7a.1 *System* suggests replacement missions to *SuperObserver*.
  7a.2 Use case continues with step 4.

7b. Mission cannot be created and no replacement missions are possible.
  7b.1 *System* suggests notifying the *NationalCrisisCenter*.
  7b.2 Use case continues with step 4.

8a. Mission failed.
  8a.1 *System* informs *SuperObserver* and *Coordinator* about mission failure.
  8a.2 Use case continues with step 4.

[1]Mission-specific information includes things such as the quantity of vehicles requested, special characteristics of the aid worker or vehicle, etc.

4.2.7 Execute Rescue Mission

Use Case 7: Execute Rescue Mission
Scope: Car Crash Crisis Management System
Primary Actor: FirstAidWorker
Secondary Actor: HospitalRS
Intention: The intention of the FirstAidWorker is to accept and then execute a rescue mission that involves transporting a victim to the most appropriate hospital.
Main Success Scenario:

FirstAidWorker is at the crisis location.

1. *FirstAidWorker* transmits injury information of victim to *System*.

Steps 2 and 3 are optional.

2. *FirstAidWorker* determines victim's identity and communicates it to *System*.
3. *System* requests victim's medical history information from all connected *HospitalResourceSystems*.

FirstAidWorker administers first aid procedures to victim.

4. *System* instructs *FirstAidWorker* to bring the victim to the most appropriate hospital.
5. *FirstAidWorker* notifies *System* that he is leaving the crisis site.
6. *FirstAidWorker* notifies *System* that he has dropped off the victim at the hospital.
7. *FirstAidWorker* informs *System* that he has completed his mission.

Use case ends in success.

Extensions:

4a. *HospitalResourceSystem* transmits victim's medical history information to *System*.

4a.1 *System* notifies *FirstAidWorker* of medical history of the victim relevant to his injury.
4a.2 Use case continues at step 4.

4.2.8 Execute Helicopter Transport Mission

Use Case 8: Execute Helicopter Transport Mission
Scope: Car Crash Crisis Management System
Primary Actor: Pilot
Secondary Actor: None
Intention: The intention of the Pilot is to accept and then execute a transport mission that involves transporting a CMSEmployee to and from a mission location.
Main Success Scenario: To be defined.

4.2.9 Execute Remove Obstacle Mission

Use Case 9: Execute Remove Obstacle Mission
Scope: Car Crash Crisis Management System
Primary Actor: TowTruckDriver
Secondary Actor: None
Intention: The intention of the TowTruckDriver is to accept and then execute a remove obstacle mission that involves removing a crashed car from a mission location.
Main Success Scenario: To be defined.

4.2.10 AuthenticateUser

Use Case 10: AuthenticateUser
Scope: Car Crash Crisis Management System
Primary Actor: None
Secondary Actor: CMSEmployee
Intention: The intention of the System is to authenticate the CMSEmployee to allow access.
Main Success Scenario:

1. *System* prompts *CMSEmployee* for login id and password.
2. *CMSEmployee* enters login id and password into *System*.
3. *System* validates the login information.

Use case ends in success.

Extensions:

2a. *CMSEmployee* cancels the authentication process. Use case ends in failure.
3a. *System* fails to authenticate the *CMSEmployee*.
3a.1 Use case continues at step 1.
3a.1a *CMSEmployee* performed three consecutive failed attempts.
3a.1a.1 Use case ends in failure.

5 Domain Model

The Domain Model offers insight into the problem domain, in our case Car Crash crisis management systems. Taking the form of a UML class diagram, it provides a description of the concepts of the problem domain relevant to the Car Crash CMS, by representing the concepts as classes, attributes and associations between classes. Although any domain concept could be added to the domain model, we decided to include here only concepts that must define information that must be recorded for the purpose of fulfilling the system's responsibilities over time. In other words, the domain model presented here only contains concepts that are used to describe the necessary information to fulfill system goals.

Because of size constraints, the domain model is split into two figures. Figure 4 depicts the Crisis and Mission concepts and how they relate to the other concepts, whereas figure 5 shows the inheritance hierarchies inherent in the domain of the Car Crash CMS.

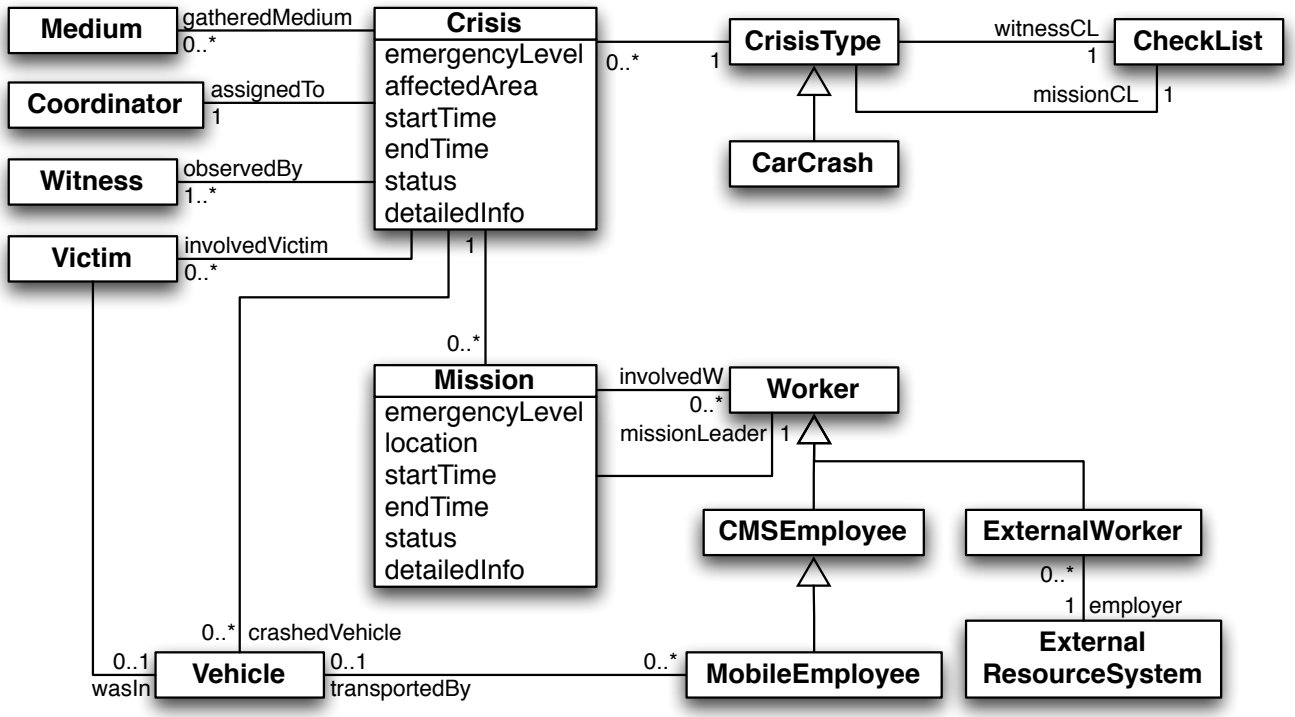

Figure 4: Car Crash CMS Domain Model, Part 1

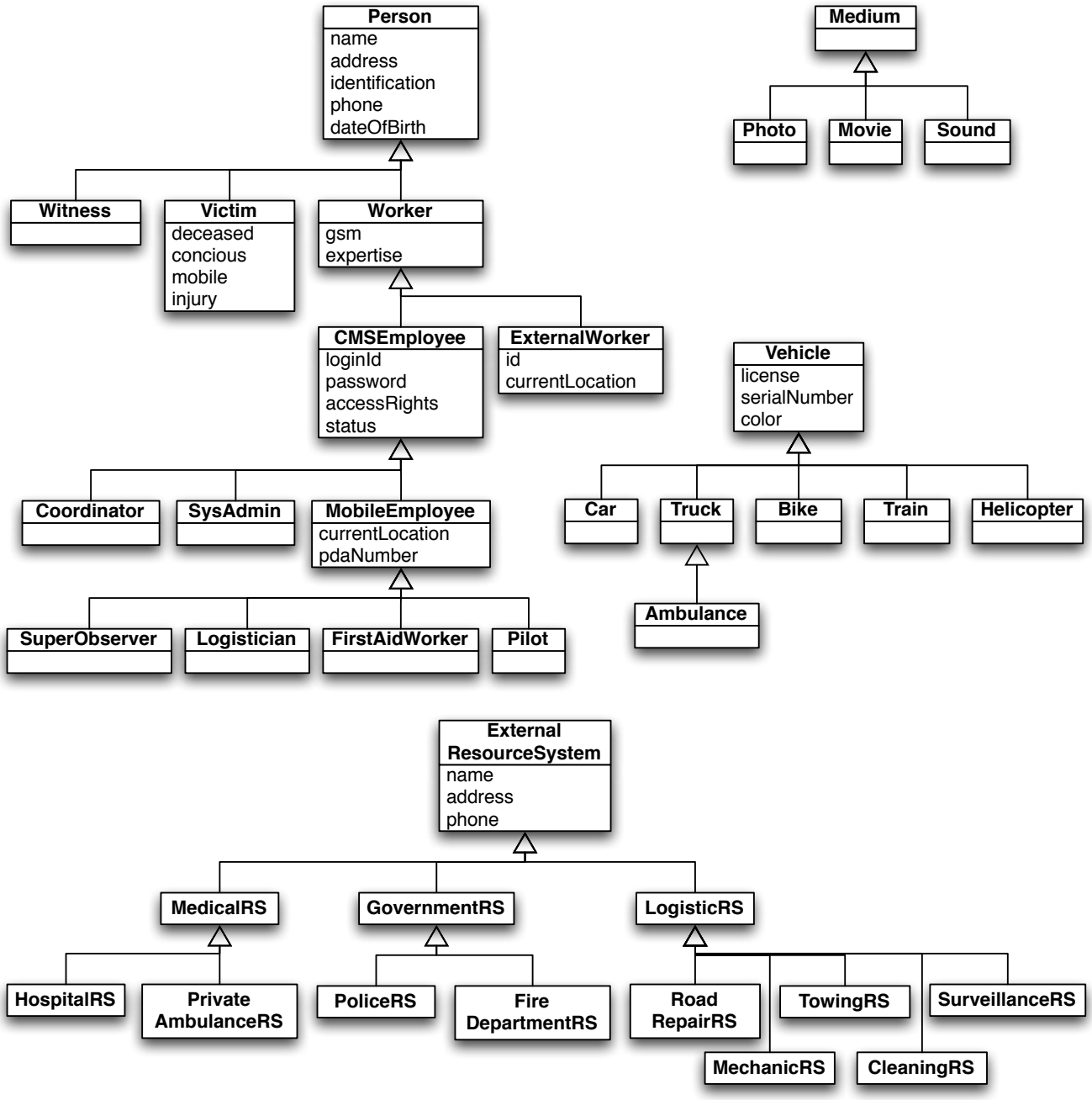

Figure 5: Car Crash CMS Domain Model, Part 2 – Inheritance Hierarchies

6 Activity Diagrams

The use cases presented in section 4 are text-based, and can hence be effectively used as a communication means between technical as well as non-technical stakeholders. In order to elaborate a more formal specification, use cases are typically refined and then depicted using sequence diagrams or activity diagrams.

In this section we give one example of a use case represented using an activity diagram. Figure 6 shows the activity diagram created based on the use case *Assign Internal Resource* already presented in subsection 4.2.3 and reproduced here for convenience. The interested reader is referred to [2] where we have defined precise transformation rules that make it possible to automate mappings from use cases to activity diagrams.

Use Case 3: AssignInternalResource
Scope: Car Crash Crisis Management System
Primary Actor: None
Secondary Actor: CMSEmployee
Intention: The intention of System is to find, contact, and assign a mission to the most appropriate available CMSEmployee.
Main Success Scenario:

System selects an appropriate CMSEmployee based on the mission type, the emergency level, location and requested expertise. In very urgent cases, steps 1 and 2 can be performed for several CMSEmployees concurrently, until one of the contacted employees accepts the mission.

1. *System* requests the *CMSEmployee* to login.
2. *System* sends *CMSEmployee* mission information.
3. *CMSEmployee* informs *System* that he accepts the mission.

Use case ends in success.

Extensions:

1a. *CMSEmployee* is already logged in.
  1a.1 Use case continues at step 2.
2a. *CMSEmployee* is unavailable or unresponsive.
  2a.1 *System* selects the next appropriate *CMSEmployee*.
  2a.2 Use case continues at step 1.
    2a.1a No other *CMSEmployee* is available. Use case ends in failure.
3a. *CMSEmployee* informs *System* that he cannot accept the mission.
  3a.1 *System* selects the next appropriate *CMSEmployee*.
  3a.2 Use case continues at step 1.
    3a.1a No other *CMSEmployee* is available. Use case ends in failure.

In the activity diagram presented in figure 6 the outcomes of the use case are modelled as *return parameters* of the activity, and hence visualized as *ActivityParameterNodes* or *output pins*. The type of outcome is identified with the predefined stereotypes $\ll success \gg$, $\ll degraded \gg$ and $\ll failure \gg$. The exceptional nature of degraded and failure outcome flows is depicted using exceptional pins.

To prevent confusion, data flow is always separated from control flow, i.e., if the activity requires input or output parameters, separate input and output pins are used.

The *AssignInternalResource* activity is composed of several subactivities beginning with *DetermineMostAppropriateEmpl*. After succcessfully selecting a CMSEmployee, the flow of *AssignInternalResource* continues to the activity *RequestLogin*. If *DetermineMostAppropriateEmpl* fails, the parent activity also fails. Next if the login is successful, it is followed by the activity *SendMissionInformation*. On a successful outcome, the system waits for a reply from the employee which is part of the activity *AwaitingMissionAcceptance*. If the login is unsuccessful or if the mission information is not transmitted successfully, the activity *AssignInternalResource* restarts by choosing an alternate worker.

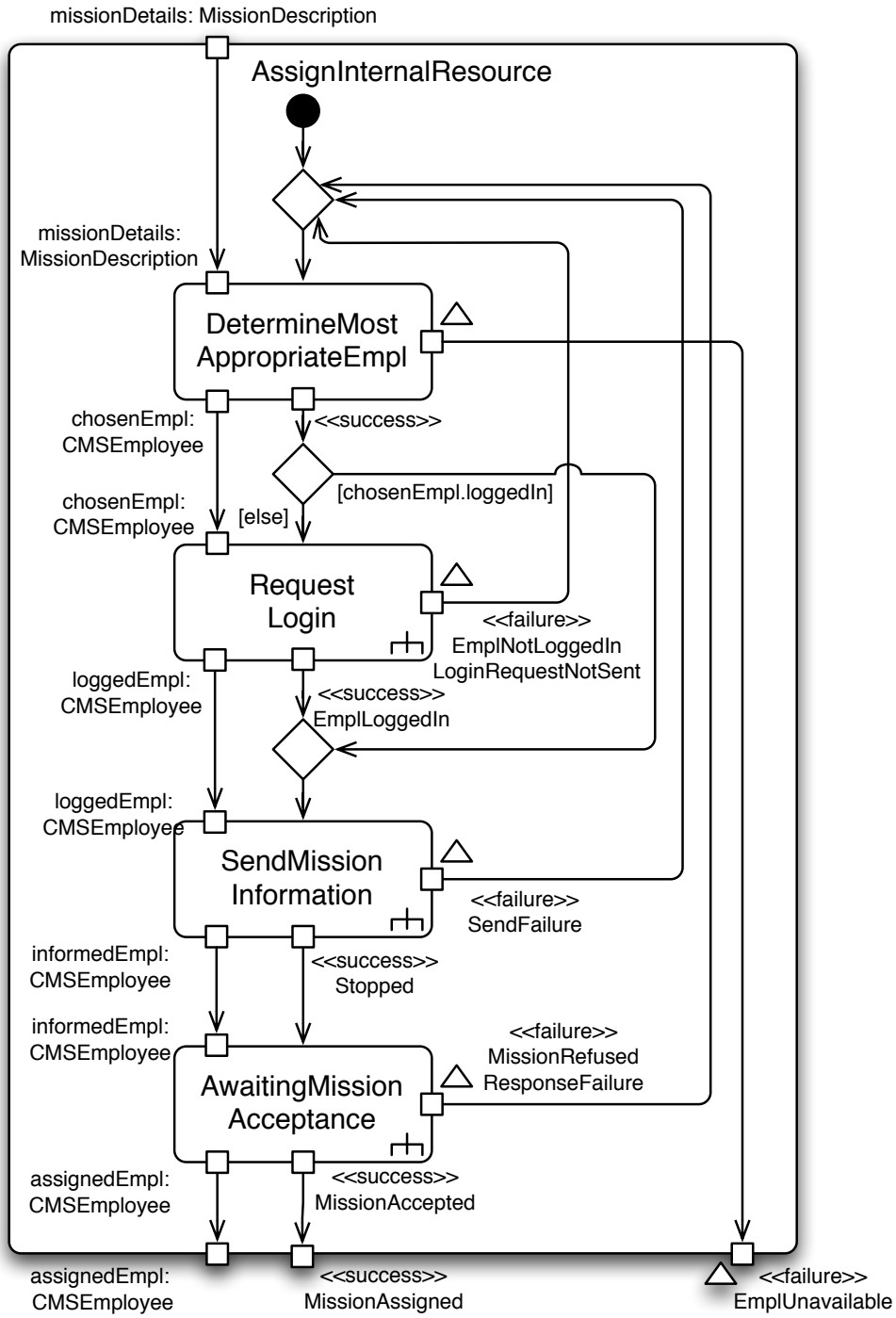

Figure 6: Car Crash Case Study: Assign Internal Resource Activity Diagram

The *AssignInternalResource* activity has 2 different outcomes. It can either succeed ($\ll success \gg$ *MissionAssigned*) or fail ($\ll failure \gg$ *EmplUnavailable*).

If desired, all use cases can be transformed into activity diagrams following the same mapping rules to create a complete specification of the system's behavior using activity diagrams.

7 Informal Physical Architecture Description

A typical architecture for a crisis management system contains many machines that are connected with different types of networks. Figure 7 gives an overview of the kinds of machines and communication networks that could be used in an instance of the Car Crash CMS.

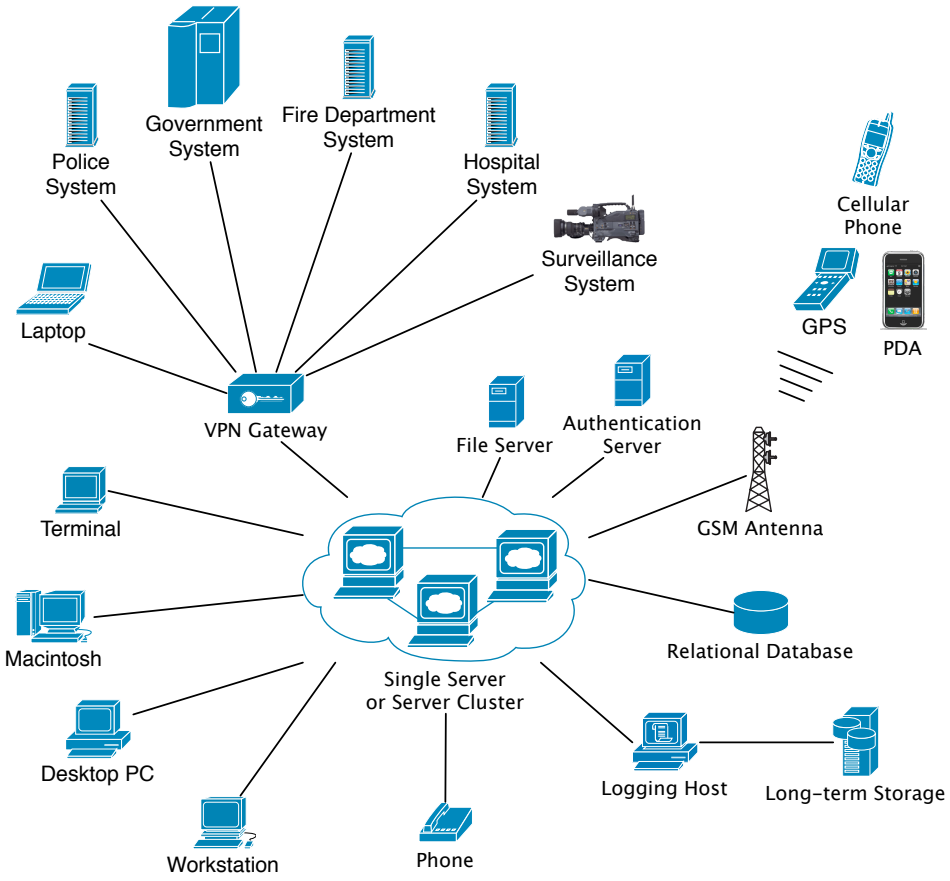

Figure 7: Car Crash Case Study: Physical Architecture

The backend of the system is composed of a server or a server cluster that implements most of the business functionality. Local CMS employees, such as the coordinators and the system administrators, use terminals or desktop machines to access the backend through a private network. External services and mobile CMS employees with laptops are connected to the backend by means of virtual private networks on top of public networks. Cell phones, GPS devices and PDAs are reached using a GSM antenna.

In the network layer, several protocols can be used to transport the communications: GSM for voice, GSM for SMS, UDP/TCP/IP for voice, and TCP/IP for data exchange. In the application layer, several protocols can be used to transport the communications: proprietary protocols or standardized protocols (HTTP, SMTP, POP3, IMAP, XMPP, etc...)

8 Selected Design Models

During design, a blue print of a solution that satisfies the requirements defined by the analysis models is devised. In object-oriented design, the conceptual state has to be mapped to objects, and then the developer has to decide how the conceptual state changes specified in every system operation are to be implemented by interacting objects at run-time.

The concepts identified during analysis, in our case e.g. Crisis, Mission, CMSEmployee, are initial candidates for becoming design objects that hold the application state. However, some concepts may be implemented using several objects, or, alternatively, some concepts may be implemented as attributes. The granularity of objects affects several aspects of the system under development. Too fine-grained decomposition leads to systems with thousands of objects. Such systems might be hard to understand and maintain due to their high coupling, and generate huge communication overhead. On the other hand, a coarse decomposition leads to bulky architectures, and objects with unclear responsibilities, which can also be hard to understand and maintain. Good designers try to maximize object coherence, while minimizing object coupling.

The idea of this section is to present some design models of a possible object-oriented design of the Car Crash CMS backend. Currently, the only functionality that is designed is the *CreateMission* functionality that is triggered by the SuperObserver when ordering missions to deal with the crash.

8.1 Creating Missions

8.1.1 Summary of Functionality

The *CreateMission* functionality allows the SuperObserver to inform the Car Crash CMS about a mission that needs to be accomplished in order to deal with the crash. The system has to store the relevant mission information, determine the candidate CMSEmployees that could accomplish the mission, establish contact with at least one of them and propose the mission to him. The design of *CreateMission* also implements some secondary functionality. For instance, it takes care of gathering statistics on how many potential candidate employees the system was able to choose from when assigning the mission. Also, it makes sure that employees have properly logged into the system (and hence authenticated) before sending them mission related information.

8.1.2 Interaction Design

It is assumed that somehow the user interface on the PDA allows the SuperObserver to select the appropriate mission kind, select the emergency level of the mission and enter detailed mission information before sending the request to the Car Crash CMS backend. The sequencing of message exchanges between objects that are triggered by this request are shown in a sequence diagram in Figure 8.

The initial request is directed to the *CrisisManager*. After instantiating a new mission object and linking it to the crisis, the crisis manager hands the responsibility of assigning the mission to a CMSEmployee to the *ResourceManager*. The resource manager has access to a hash table of employees indexed by expertise, and hence is able to obtain a collection of employees that are qualified to execute the mission. The resource manager then proceeds by looping though this list, and inserting any available employee (i.e. an employee that currently is not affected to other missions or otherwise unavailable) that is close enough to the mission location (i.e. can get to the mission location in a reasonable amount of time) into a priority queue. In the queue, the employees are sorted with respect to their ”adequacy” for performing the mission. Once this list is established, the size of the list is remembered for statistical

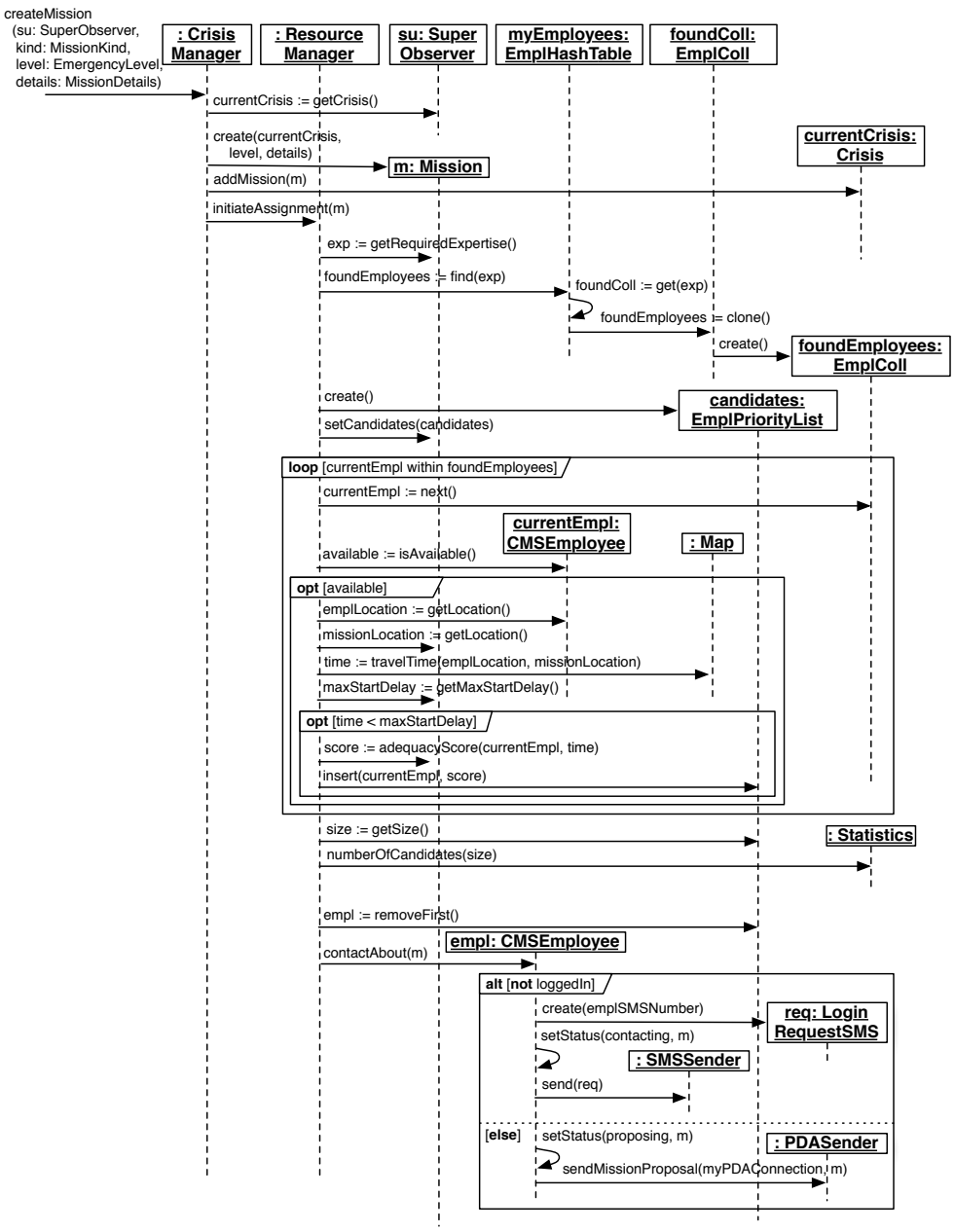

Figure 8: Car Crash Case Study: CreateMission Design Sequence Diagram

purpose. Finally, the resource manager proceeds by contacting the first employee on the list. If that employee is not currently logged in, then he is requested to do so by sending him an SMS.

The interaction design ends here, because the system now has to wait for an answer from the employee (which can either be a login request, or a mission acceptance notification).

8.1.3 Structural Design

The chosen design solution presented in the sequence diagram has many implications on the design. It assumes, for instance, the existence of many classes with particular fields and method definitions. Also, some permanent associations are assumed to be existing. For example, a super observer must be

associated with a crisis. This is obvious from the first message in the sequence diagram in Figure 8. Another example is the employee hash table that can find employees based on a particular expertise. The *CreateMission* design assumes that this hash table already exists, which means that the functionality that deals with the creation of employees must also take care of building this hash table, which established permanent references between expertise and groups of employees.

The classes, attributes and methods, and the dependencies and associations between classes created, used and assumed by the *CreateMission* design are shown in Figure 9.

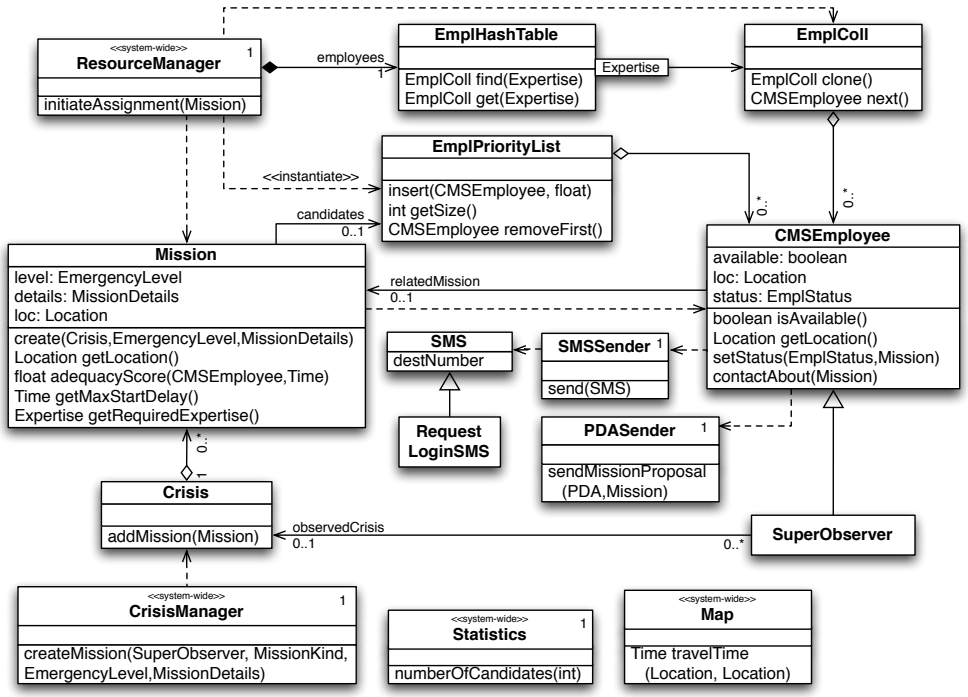

Figure 9: Car Crash Case Study: Partial Design Class Diagram based on CreateMission Design

References

[1] Sadaf Mustafiz and Jörg Kienzle. A Requirements Engineering Process for Dependable Reactive Systems. In *Methods, Models and Tools for Fault Tolerance*.

[2] Sadaf Mustafiz, Jörg Kienzle, and Hans Vangheluwe. Model transformation of dependability-focused models. In *MiSE '09: Proceedings of the 2009 International Workshop on Models in Software Engineering*, New York, NY, USA, 2009. ACM. To be published.

[3] Optimal Security. Product line document: Version 0.7. March 2009.

[4] Optimal Security. Requirements document: Version 0.8. March 2009.