Laboratory for Advanced Software Systems
University of Luxembourg
6, rue R. Coudenhove-Kalergi, Luxembourg
with participation of:
Vicious & Delicious Working Group
Heidi Flinkman, Alexander Eyjolfsson, Yan Medernach

iCrash :
A Crisis Management Case Study
Messir Analysis Document
- v 1.8 -

Tuesday 8th April, 2014 - 22:51

Springer

Contents

List of Figures

Listings

Chapter 1
Introduction

1.1 Overview

iCrash is a simple system dedicated to any person who wants to inform of a car crash crisis situation in order to allow for crisis handling.
At anytime and anywhere, anyone can be the witness or victim of a car crash and might be in a situation allowing for alerting this crisis.
The *iCrash* system has for objectives to support crisis declaration and secure administration and crisis handling by the *iCrash* professional users.

1.2 Purpose and recipients of the document

This document is an analysis document complying with the Μεσσιρ methodology [1]. Its intent is to provide an example of a precise specification of the functional properties of the *iCrash* system.

The recipients of this document are:

- the *iCrash* system's buyer company (ABC): this document is used as a contractual document jointly with any other document considered as useful (as requirement elicitation document, ...) in order to have a higher degree of precision in requirement description. It is also used as a basis document for the *iCrash* system validation using specification based testing.
- the *iCrash* system development company (ADC) is expected to use this document as the basis for development (mainly design, implementation, maintenance). It is also used for verification and validation using test plans defined using the analysis models described in this document and according to the Μεσσιρ methodology.

1.3 Application Domain

The *iCrash* system belongs to the Crisis Management Systems Domain. It is a system dedicated to crisis professional and non professional end users. It has to be considered as an autonomous and external service for the society. It is not an institutional system certified and guaranteed by any governmental entity and thus, must be used with caution.

1.4 Definitions, acronyms and abbreviations

N.A.

1.5 Document structure

The document structure is designed to be coherent with the Messir methodology [1]. Section 2 provides a general description of the system purpose, its users, its environment and some general non functional requirements. A more detailed description of the non functional requirements, if any, are provided in section 7.
The system operation triggered by events sent by the external *actor*s belonging to the environment are described in Section 3. The *iCrash* concepts used to represent the any persistent or transient information is given in Section 4. The precise specification of the system operations in term of system's state changes, events sent together with the constraints on the allowed sequences of system operations are described in Section 5.

Chapter 2
General Description

In the context of the Messir method, the information provided in this section is intended to present the system for which the Messir analysis is provided. The content of this section is made accordingly to the requirements elicitation document that might have been done during the project but also adapted coherently in order to be an abstract introduction to the Messir analysis.

2.1 Domain Stakeholders

All stakeholders of the system are detailed in this section. After a brief description of a stakeholder, its objectives are first stated. Thereafter, the responsibilities of the stakeholder are detailed which help to achieve the stakeholder objectives to a certain degree. While the objectives characterize the general problems addressed by the *iCrash* system, the responsibilities describe concrete actions that are expected from a stakeholder. Some of these responsibilities can be traced looking at the use case described in Section 2.3,and hence must be supported by the *iCrash* system. All stakeholders listed in this section have an interest in the system or are affected by the system in some way, but only a subset of the stakeholders are directly involved in the use cases described. Let us remind that use case diagrams or descriptions are not Messir analysis phase mandatory outputs. They are proposed as informal means to help understanding the semantics of the system specification made of the mandatory analysis models, which provide a complete executable specification.

2.1.1 Communication Company

A Communication Company is a company that has the capacity to ensure communication of information between its customers and the *iCrash* system. The objectives of a Communication Company are:

- to be able to deliver any SMS sent by any human to the *iCrash* 's phone number.
- to be able to transmit SMS messages from the ABC company that owns the *iCrash* system to any human having a SMS compatible device accessible using a phone number.

In order to achieve these objectives, the responsibilities of a Communication Company are:

- ensure confidentiality and integrity of the information sent by a human to the *iCrash* system or from the system to a human.
- to be always available and reliable.

2.1.2 Humans

A human is any person who considers himself related to a car crash either as a witness, a victim or an anonymous person. The objectives of a human are:

- inform the *iCrash* system about the crisis situation he detected.
- be sure that the ABC company has been informed about the situation.
- to be informed about the situation of the crisis he his related to as a victim or witness.

In order to achieve these objectives, the responsibilities of a human are:

- to provide has much details as possible, even tho these details might contain errors, concerning the crisis to the ABC company.
- to declare a crisis only if the crisis is real.
- to have access to the SMS compatible communication device he used to communication with the *iCrash* system.

2.1.3 Coordinators

A coordinator is a employee of the ABC company being responsible of handling one or several crisis. The objectives of a coordinator are:

- to securely monitor the existing alerts and crisis.
- to securely manage alerts and crisis until their termination.

In order to achieve these objectives, the responsibilities of a coordinator are:

- to be capable to determine how an alert received should be considered.
- to be available to react to requests to handle alerts and crisis.
- to be autonomous in handling crisis and to report on its handling.
- to be able to decide when a crisis or an alert can be closed.
- to know its system identification information for secure usage of the system.

2.1.4 Administrator

An administrator is a employee of the ABC company being responsible of administrating the *iCrash* system. The objectives of a coordinator are:

- to add or delete coordinator actors from the system and its environment.

In order to achieve these objectives, the responsibilities of a coordinator are:

- know the company employees that can be coordinators and that have access to the system.
- to know its system identification information for secure usage of the system.
- to know the security policy of the ABC company.
- to communicate the coordinators their identification information for secure system usage.

2.1.5 Creator

A `Creator`[1] is a technician who is installing the *iCrash* system on the infrastructure of the ABC company. The objectives of a `Creator` are:

- to install the *iCrash* system
- to define the values for the initial system's state

[1] In Μεσσιρ each system has a creator stakeholder who is different from a technical system administrator. Although those two roles can be fulfilled by the same person.

- to define the values for the initial system's environment
- to ensure the integration of the *iCrash* system with its initial environment

In order to achieve these objectives, the responsibilities of a `Creator` are:

- provide the necessary data to the *iCrash* system for its initialization.

2.1.6 Activator

An `activator` is a logical representation of the active part the *iCrash* system. It represents an implicit stakeholder belonging to the system's environment that interacts with the *iCrash* system autonomously without the need of a external entity. It is usually used for representing time triggered functionalities.

The objectives of a `activator` are:

- to communicate the current time to the system
- to notify the administrator that some crisis are still pending for a too long time.

In order to achieve these objectives, the responsibilities of a `activator` are:

- to know the current universal time
- to send the messages to the system according to the time constraints specifically defined for it.

2.2 System's Actors

The objective of this section is not to provide the full requirement elicitation document in this section but to reuse a part of this document to provide a informal introduction to the Messir specification of the system under development. The use case model is made of a use case diagrams modelling abstractly and informally the actors and their use cases together with a set of use cases descriptions. In addition, those diagrams and description tables are adapted to the Messir specification since actor and messages names together with parameters are partly adapted to be consistent with the specification identifiers (see [1] for more details).

Among all the stakeholders presented in the previous section, we can determine five types of *direct actors*[2]:

- `actComCompany`: for the Communication Company stakeholder.
- `actAdministrator`: for the Administrator stakeholder.
- `actCoordinator`: for the Coordinators stakeholders.
- `actActivator`: for the Activator stakeholder.
- `actMsrCreator`: for the Creator stakeholder.

In addition to those system actors, we can add five other types of actors related to the system's ones. Those five actors are grouped into two categories:

- *Indirect actors*
  - *Witness*: for any human that is a witness of a car crash
  - *Victim*: for any human that is a victim of a car crash
  - *Anonymous*: for any human that want to inform about a car crash while staying anonymous.
- *Abstract actors*
  - `actHuman`: represent abstractly any kind of human being actor wanting to communicate with the ABC system in the context of a car crash.
  - `actAuthenticated`: for the logical Activator stakeholder.

[2] The naming conventions in Messir propose to start each type name by lowercase letters indicating the meta model type used (i.e. act for actors, ct for class type, ...). In addition to ease the reading it makes the translational semantics into Prolog code more straightforward.

2.3 Use Cases Model

2.3.1 Use Case Diagram(s)

The figures 2.1, 2.2, 2.3, 2.4 and 2.5 provide Messir use casediagrams for the ABC system use cases.

Even though this general description is not formal, follow the advice given in the Messir book [1] which proposes to have for each actor an input interface to receive messages from the system and an output one to send messages to the system.

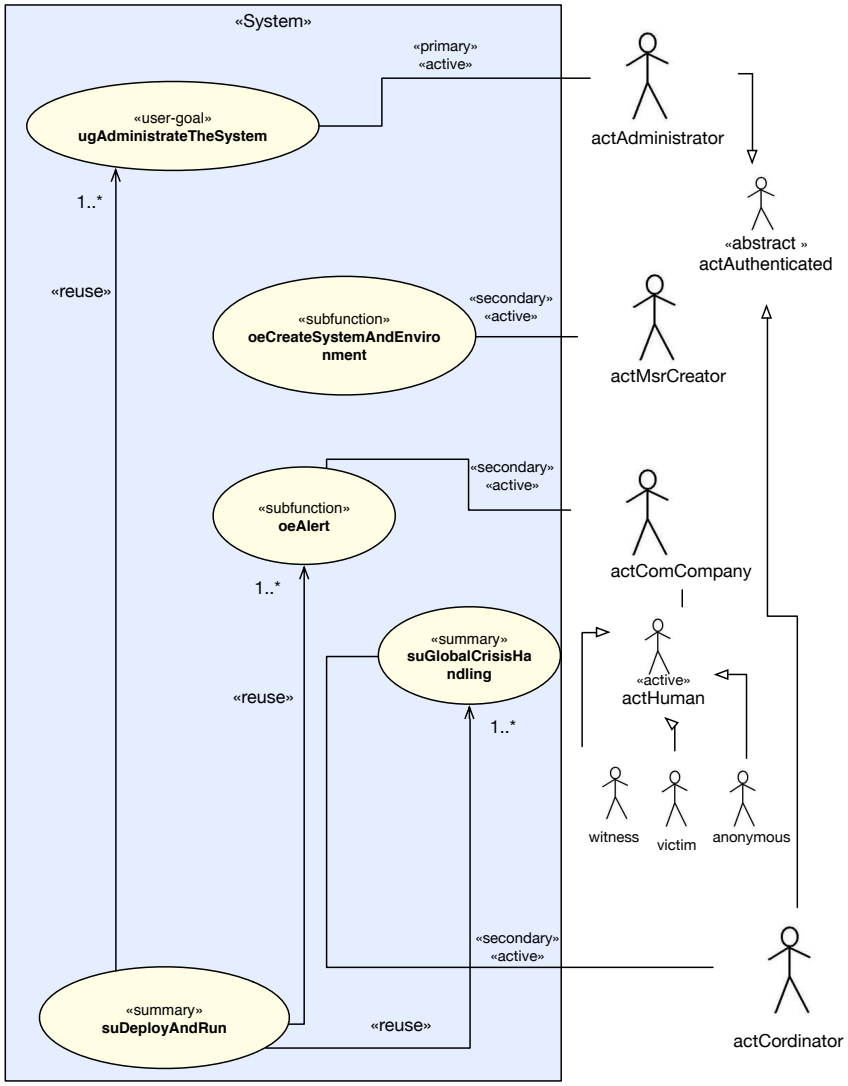


Fig. 2.1 *iCrash* suDeployAndRun Use Case Diagram

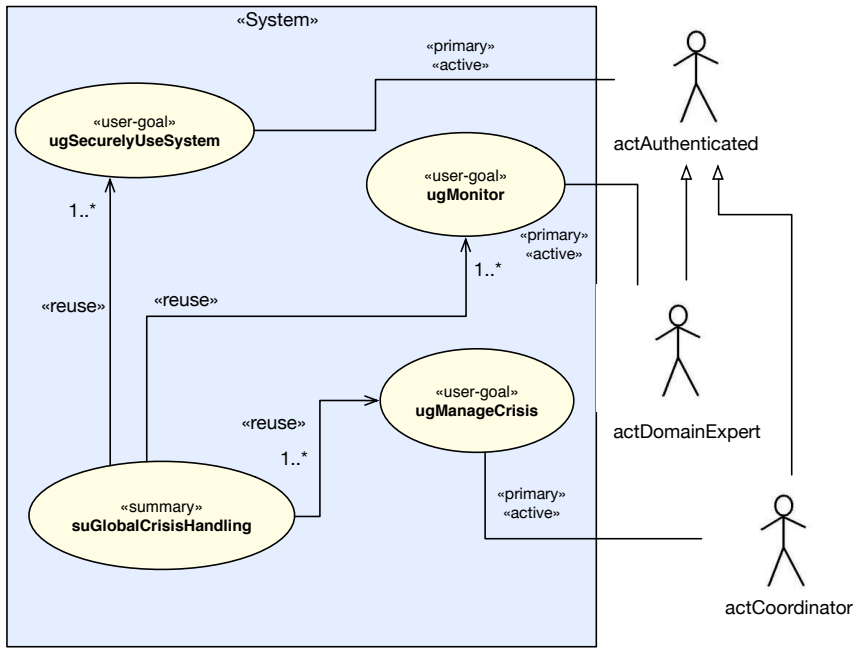


Fig. 2.2 *iCrash* suGlobalCrisisHandling Use Case Diagram

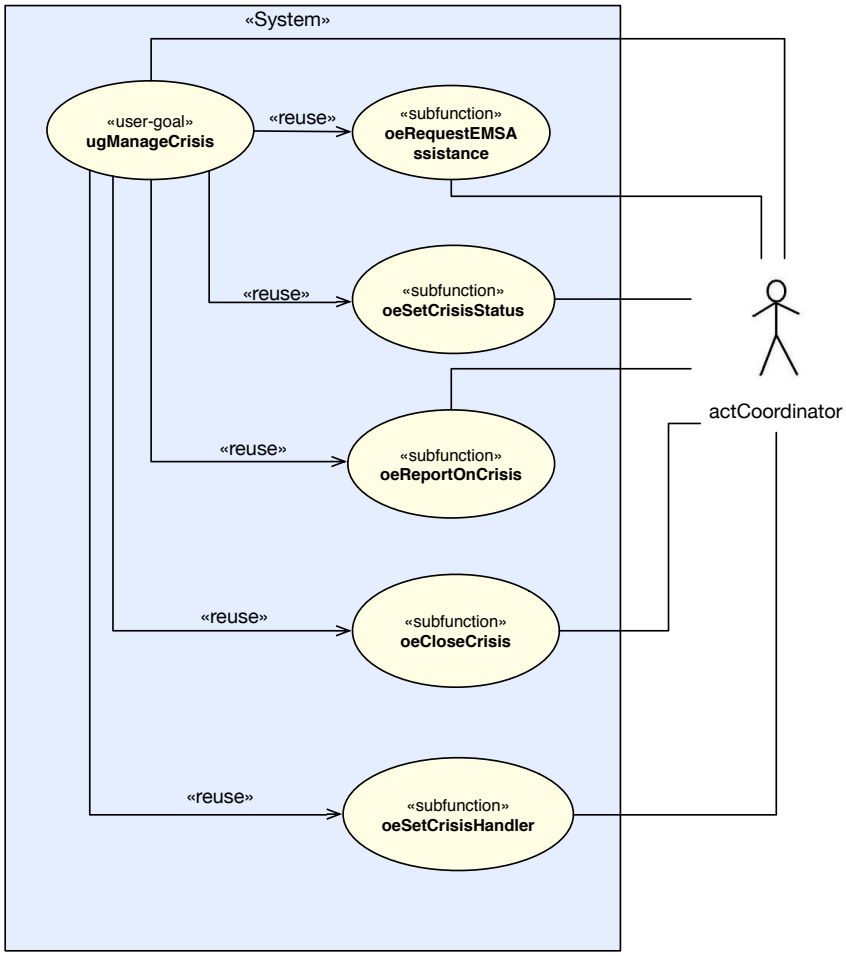


Fig. 2.3 *iCrash* ugManageCrisis Use Case Diagram

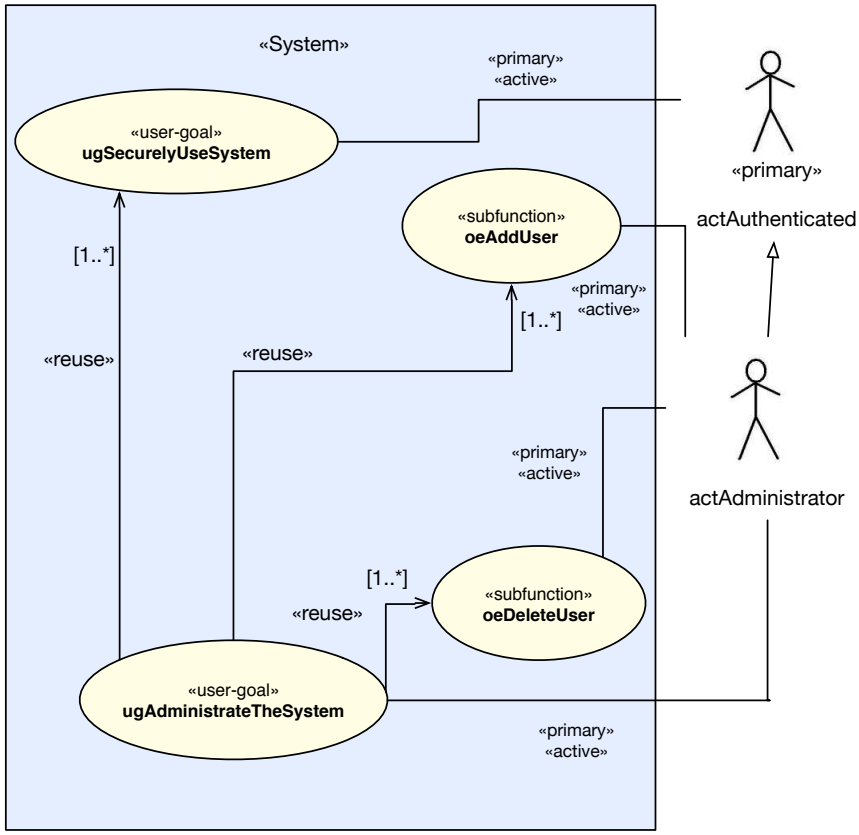


Fig. 2.4 *iCrash* ugAdministrateTheSystem Use Case Diagram

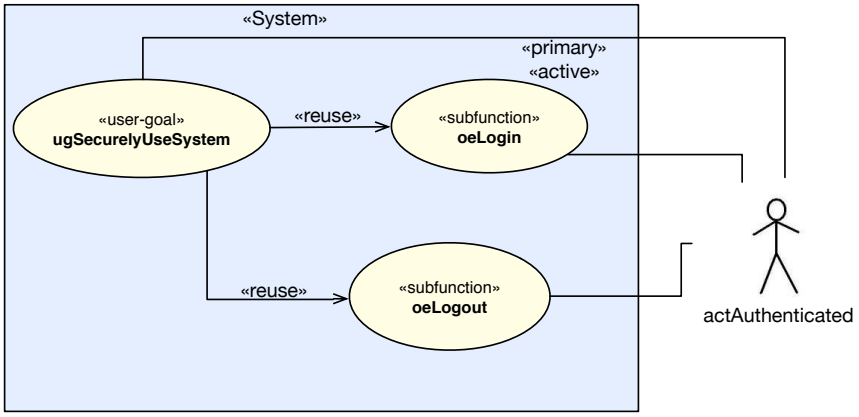


Fig. 2.5 *iCrash* ugSecurelyUseSystem Use Case Diagram

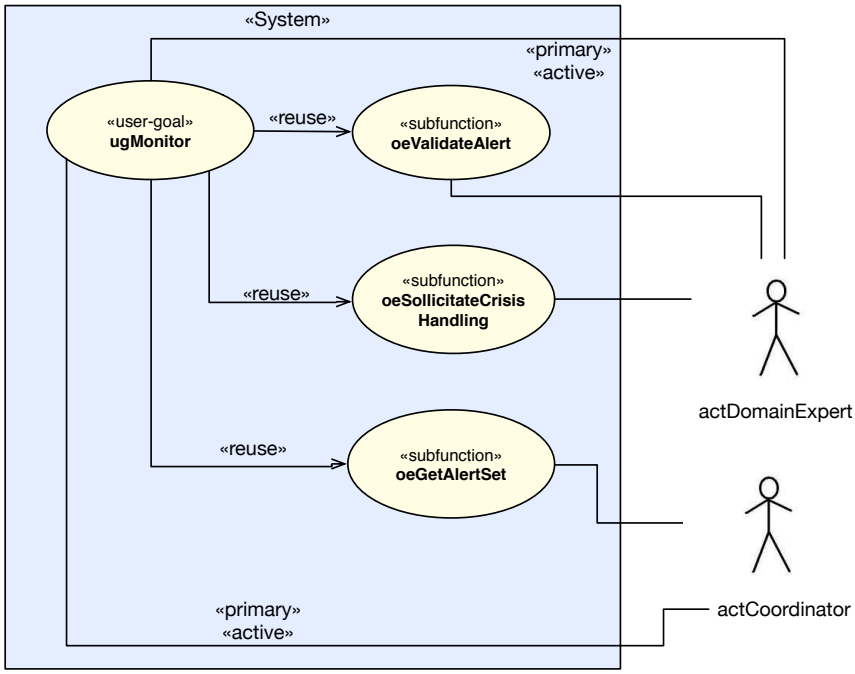


Fig. 2.6 *iCrash* ugMonitor Use Case Diagram

2.3.2 Use Case Instance(s)

The use case instance described below represents a summary level use case instance intended to illustrate a small and complete execution scenario of the *iCrash* system. Figure A.24 provides informal graphical view (using a sequence diagram) of the messages sent during the scenario. The use case instance is described textually in the following use case description form as suggested by the Messir method and inspired by the standard Cokburn template [2]. We chose to indicate only messages sent by the environment to the system and not reply messages for readability.

Use-Case Instance

Name
Simple and Complete DeployAndRun Instance

Instance ID
01

Environmental conditions and assumptions
The necessary IT infrastructure exists to allow for deployment of the *iCrash* system.

Inputs
inputs are sequences of characters interpreted as string or numbers.

Instance flow description

1 the `actMsrCreator` actor `theCreator` sends the message `oeCreateSystemAndEnvironment(1)` requesting the initialization of the system and its environment (consisting of one administrator identified here by `bill`) and indicating that the number of communication company actor instances for the system's environment is 1 (identified here by `tango`).

2 the `actAdministrator` actor `bill` sends the message `oeLogin(icrashadmin,7WXC1359)` to securely connect to the system.

3 the `actAdministrator` actor `bill` sends the message `oeAddUser(1,Steve, pwdMessirExcalibur2017,1635242A75, 'Vehicular, Pedestrian, Wildlife, Property, Injury',Coordinator)` to set up a coordinator (i.e. identified here by `steve`) and indicating his identification information, ID (i.e. `1`) and a password (i.e. `pwdMessirExcalibur2017`), id from a token(i.e.`1635242A75`), specifying his domain of expertise(i.e. `'Vehicular, Pedestrian, Wildlife, Property, Injury'`) and finally he designates the user as a Coordinator(i.e.`Cooridnator`).

4 the `actAdministrator` actor `bill` sends the message `oeAddUser(2, franklin, pwdMessirExcalibur, 1456872B82, Null, DomainExpert)` create a domain expert who is one person(i.e. identified here by `franklin`) and indicating his user identification his identification, ID(i.e.`2`), password(i.e.`pwdMessirExcalibur123`), id from a token(i.e.`14566872B82`), specifying the users domain of expertise as(`Null` meaning that the user has no specific domain and can therefore only be an EMS user or Domain expert) and finally he designates the user as a domain expert(i.e.`DomainExpert`).

5 the `actAdministrator` actor `bill` sends the message `oeAddUser(3, barry, pwdEMSExcalibur321, 2436787C82, Null, EMS)` to set up a EMS user who is any user in the EMS headquarters(i.e. identified here by `barry`) and indicating his user identification as ID(i.e.`3`), password (i.e.`pwdEMSExcalibur421`), id from a token(i.e.`2436787C82`), specifying the users domain of expertise as(i.e.`Null` meaning that the user has no specific domain and can therefore only be an EMS user or DomainExpert) and finally he designates the user as an EMS user(i.e.`EMS`).

6 the `actAdministrator` actor `bill` sends the message `oeLogout()` to disconnect from the system.

7 the `actComCompany` actor `tango` sends the message `oeAlert(witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590, '3 cars involved in an accident.')` to transfer a declaration of a car crash by a witness indicating specific phone number, the date and time, the GPS coordinates of the witnessed car crash and a short message giving additional details.

8 the `actDomainExpert`actor `franklin` sends the message `oeLogin(franklin, pwdMessirExcalibur123, 687594FAD9)`to securely connect to the system entering his login(i.e.`franklin`), his password(i.e.`pwdMessirExcalibur123`)and entering his serial key that he reads form a token device given to him by the administrator(i.e.687594FAD9).

9 the `actEMS`actor `barry` sends the message `oeLogin(barry, pwdEMSExcalibur321, 24367872C282)`to securely connect to the system entering his login(i.e.`barry`), his password(i.e.`pwdEMSExcalibur321`)and entering his serial key that he reads form a token device given to him by the administrator(i.e.`24367872C282`).

10 the `actDomainExpert` actor `franklin` sends the message `oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')`to validate the crisis and to set a Domain to it.

11 the `actDomainExpert` actor `franklin` sends the message `oeSollicitateCrisisHandling()` indicating that there is a declared alert that is still not handled by any coordinator.

continues in next page . . .

. . . Use-Case Instance table continuation

12 the actCoordinator actor steve sends the message oeLogin(steve, pwdMessirExcalibur2017, 63524275D9) to securely connect to the system, entering his login(i.e.steve), his password(i.e.pwdMessirExcalibur2017) and his serial Key that he reads form a token device he is given by the administrator(i.e.63524275D9)

13 the actCoordinator actor steve sends the message oeGetAlertsSet(valid) to receive information about all the pending crisis.

14 the actCoordinator actor steve sends the message oeSetCrisisHandler(1,Medium, in-handling, 49.627095-6.160251, 2017-11-26-at-10-10-16AM) to declare that he is taking care of the alert with the ID equal to 1 which becomes the Crisis Id, sets the crisis type to(i.e.Medium, the crisis type to(i.e.in-handling), enters the GPS coordinates and the date and time.

15 the actComCompany actor tango sends the message oeAlert(witness, 2017-11-26-at-10-20-18AM,+3524666445314, 49.627095-6.160251,Please send rescue services.) to transfer a declaration of a car crash by a witness indicating specific the phone number, the date and time, the GPS coordinates of the witnessed car crash and a short message giving additional details. This alert's GPS coordinates match the previous alert sent coordinates in step 7 and the time elapsed between the two alerts is 10 minutes therefore the alert is considered to be originating from the same location.

16 the actDomainExpert actor franklin sends the message oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')to validate the crisis and to set a Domain to it. Because the alert originated from the same location and only 10 min later it an automated message informs the Domain Expert to validate it to the same alert ID as the previous alert.

17 the actCoordinator actor steve send the message oeReportOnCrisis(1, ' 2 Alerts received about a car accident at the same location apparently involving 3 vehicles', 2, Medium, Handled,'witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590,'3 cars involved in an accident', witness, 2017-11-26-at-10-2-18AM,+3524666445314, 49.627095-6.160251,'Please send rescue services.',3, 3) indicating the crisis ID(i.e.1), entering a specific comment in the comment area(i.e.'2Alertsrecived about a car accident at the same location apparently involving 3 vehicles'), specifying his user ID(i.e.2), setting the CrisisType(i.e.Medium), indicating the current crisisStatus(handled), entering the previous received alerts, indicating the number of vehicles involved in the accident(i.e.3) and entering the number of victims(i.e.3) because he suspects that there may be 3 victims due to the amount of cars involved in the accident. Entering a preliminary report on the crisis with the information that he presumes are right.

18 the actCoordinator actor steve send the message oeRequestEMSAssistance('We have received an Alert of an accident involving 3 cars from 1 victim and 1 witness at the same location please send police and ambulance',1, 49.627095-6.160251, 3,3, Ambulance Police), entering a comment(i.e. ' We have received an Alert of an accident involving 3 cars from 1 victim and 1 witness at the same location please send Police assistance'), indicating the RequestID(i.e.1), entering the GPS location of the accident(i.e.49.627095-6.160251), informing them of the number of cars involved(i.e.3) and informing them of the presumed number of victims(i.e.3) requesting emergency services assistance(i.e.Ambulance, Police).

19 the actCooordinator actor steve sends the message oeSetCrisisStatus(1, handled)to change the status of the crisis identified by the ID(i.e.1) to the status(i.e.handled) thus indicating that he has handled the situation for now but it's not solved yet.

20 the actEMS actor barry sends the message oeReplyToRequest(1,'Message received, dispatching police and ambulance units.', 1, Handled) indicating the CrisisID(i.e.1), sending the comment(i.e.'Message received, dispatching police and ambulance units.'), to confirm that the request with the ID(i.e.1) has been received and is being handled, indicated by the EMS crisis status(i.e. Handled).

continues in next page . . .

. . . Use-Case Instance table continuation

21 the actEMS actor barry sends the message oeReportEMSCrisisStatus(1, 'Units arrived on scene report 4 victims, 3 cars involved in accident. Situation under control 1 victim brought Mercy hospital', solved, Mercy Hospital, Jeremy Springer, John Snow, +32168432) indicating the crisis ID(i.e.1), adding a comment(i.e. 'Units arrived on scene report 4 victims, 3 cars involved in accident. Situation under control 1 brought to Mercy hospital'), specifying the EMS crisis status as solved (i.e.solved), indicating the hospital name as (i.e.Mercy Hospital), indicating the victim name(i.e.Jeremy Springer), indicating the victims ICE contact's name as(i.e.John Snow) and the contacts phone number as (+32168432) to report on the EMS crisis Status.

22 the actComCompany actor tango sends message oeInfoFam('Dear John Snow, I regret to inform you that Jeremy Springer was in a car accident involving 3 other cars. Jeremy Springer was brought to the Mercy hospital for examination. Regretfully yours..',Jeremy Springer, +32168432, Mercy Hospital) to inform the victims family members about the state of the victim and in which hospital they reside.

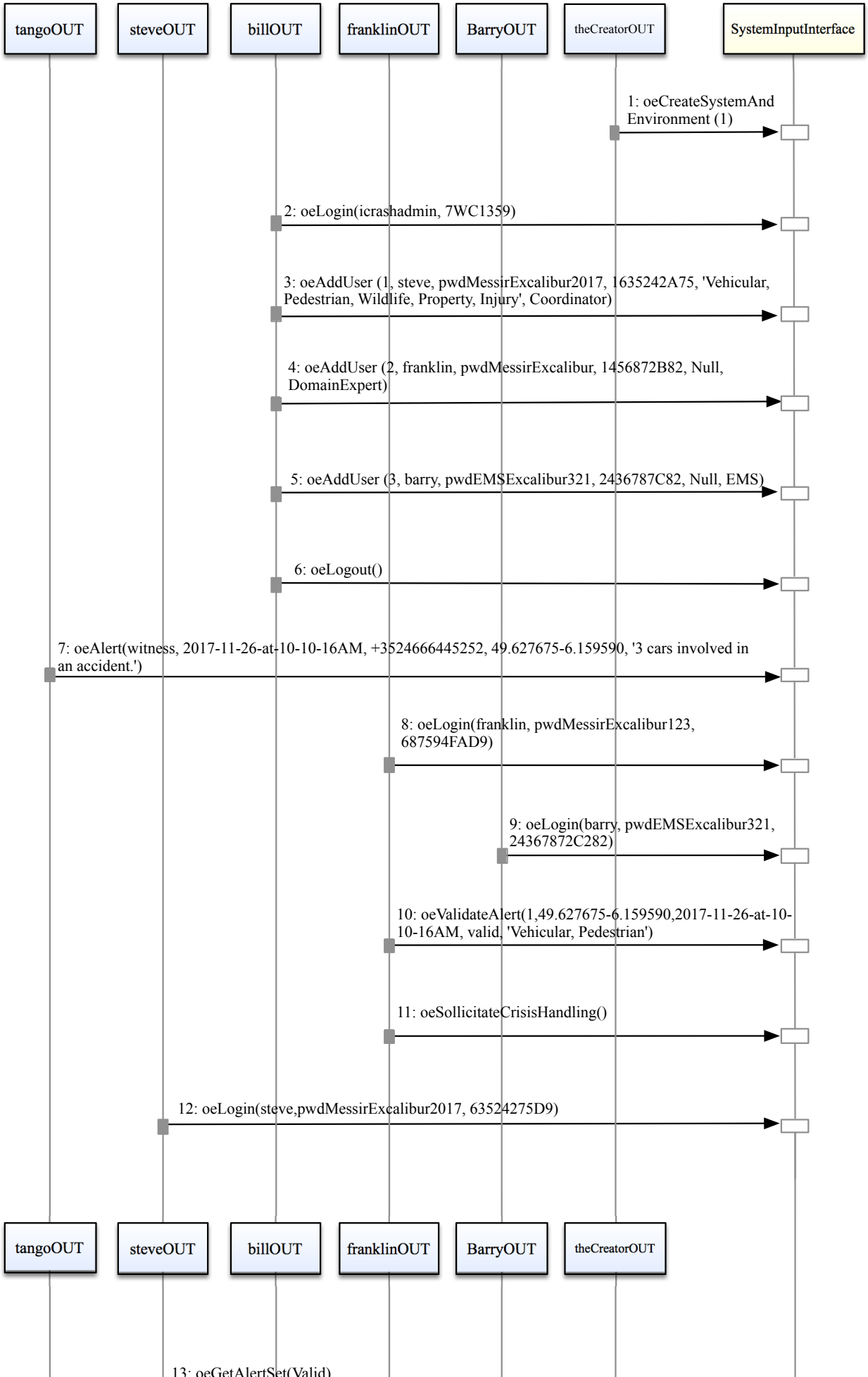
23 the actCoordinator actor steve sends the message oeReportOnCrisis(1, ' 2 Alerts received about a car accident at the same location involving 3 vehicles and 4 victims one victim Jeremy Springer was brought to the Mercy hospital.', 2, Medium, Handled,'witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590,'3 cars involved in an accident 'witness, 2017-11-26-at-10-2-18AM,+3524666445314, 49.627095-6.160251,'Please send rescue services.",3, 4) to set the report for the crisis with ID equal to 1 that he is handling.

24 the actCoordinator actor steve sends the message oeReportOnCrisis(1,'3 victims sent to hospital, 2 cars evacuated and 4 rescue unit mobilized') to set the report for the crisis with ID equal to 1 that he is handling.

25 the actCoordinator actor steve sends the message oeCloseCrisis(1) to declare the crisis with ID equal to 1 as closed.

Outputs

at the end of this instance flow the system should have its environment made of one communication company actor instance, one coordinator actor instance, one administrator instance and the creator instance. The system state should contain two alerts defined according to the information received from the communication company, only one crisis that should be considered as solved and being alerted by the two alerts received.


tangoOUT
steveOUT
billOUT
franklinOUT
BarryOUT
theCreatorOUT
SystemInputInterface
1: oeCreateSystemAnd Environment (1)
2: oeLogin(icrashadmin, 7WC1359)
3: oeAddUser (1, steve, pwdMessirExcalibur2017, 1635242A75, 'Vehicular, Pedestrian, Wildlife, Property, Injury', Coordinator)
4: oeAddUser (2, franklin, pwdMessirExcalibur, 1456872B82, Null, DomainExpert)
5: oeAddUser (3, barry, pwdEMSExcalibur321, 2436787C82, Null, EMS)
6: oeLogout()
7: oeAlert(witness, 2017-11-26-at-10-10-16AM, +3524666445252, 49.627675-6.159590, '3 cars involved in an accident.')
8: oeLogin(franklin, pwdMessirExcalibur123, 687594FAD9)
9: oeLogin(barry, pwdEMSExcalibur321, 24367872C282)
10: oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')
11: oeSollicitateCrisisHandling()
12: oeLogin(steve,pwdMessirExcalibur2017, 63524275D9)
tangoOUT
steveOUT
billOUT
franklinOUT
BarryOUT
theCreatorOUT

Chapter 3
Environment Model

We provide below the view(s) defined for the Messir environment model (cf. [1]) of the system.

3.1 Local view 01

Figure 3.1 Shows the overview of the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

3.2 Local view 02

Figure 3.2 Shows the overview of the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

3.3 Local view 03

Figure 3.3

3.4 Local view 04

Figure 3.4

3.5 Local view 05

Figure 3.5

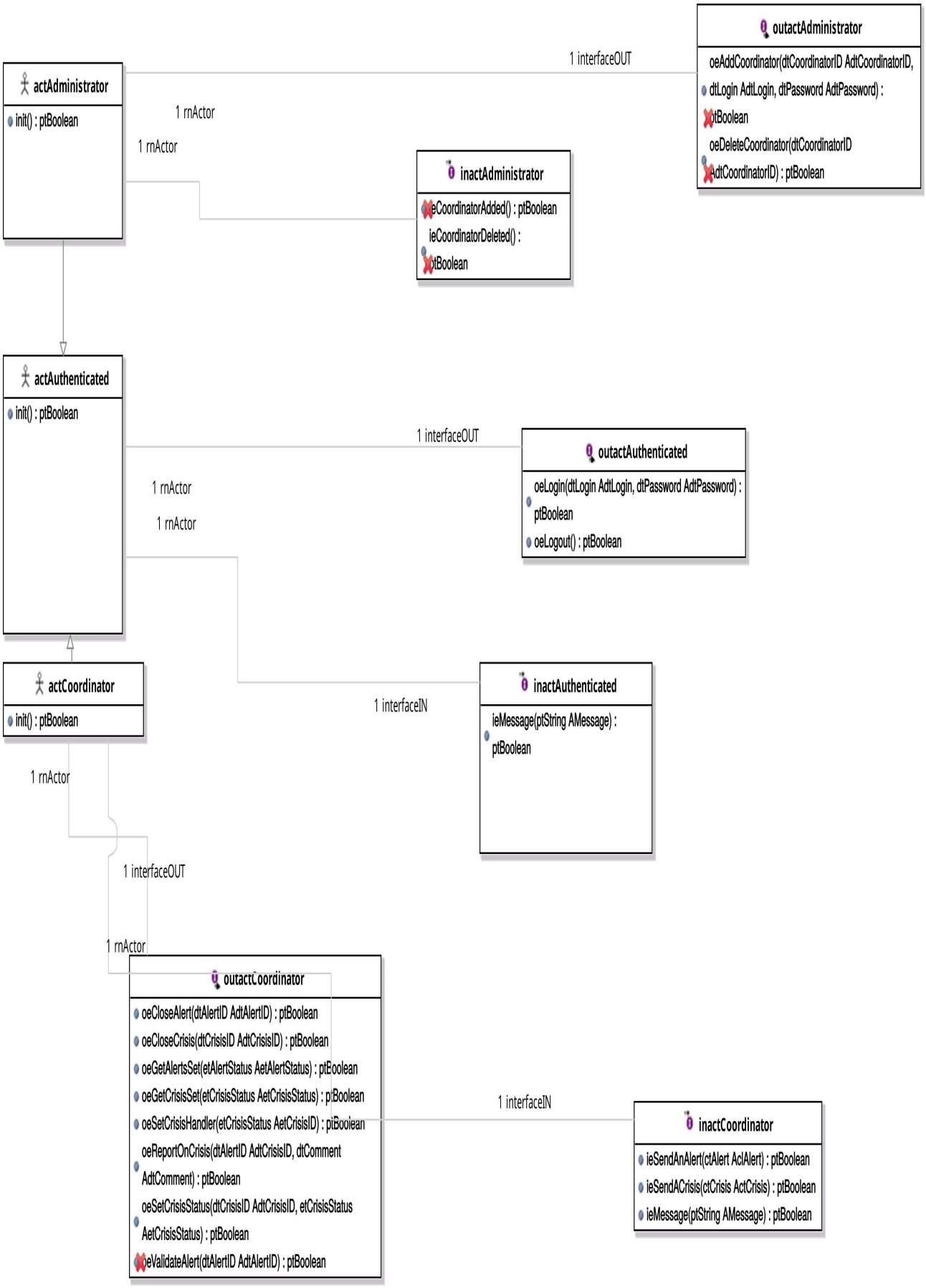


Fig. 3.1 Environment Model - Local View 01. Part 1.

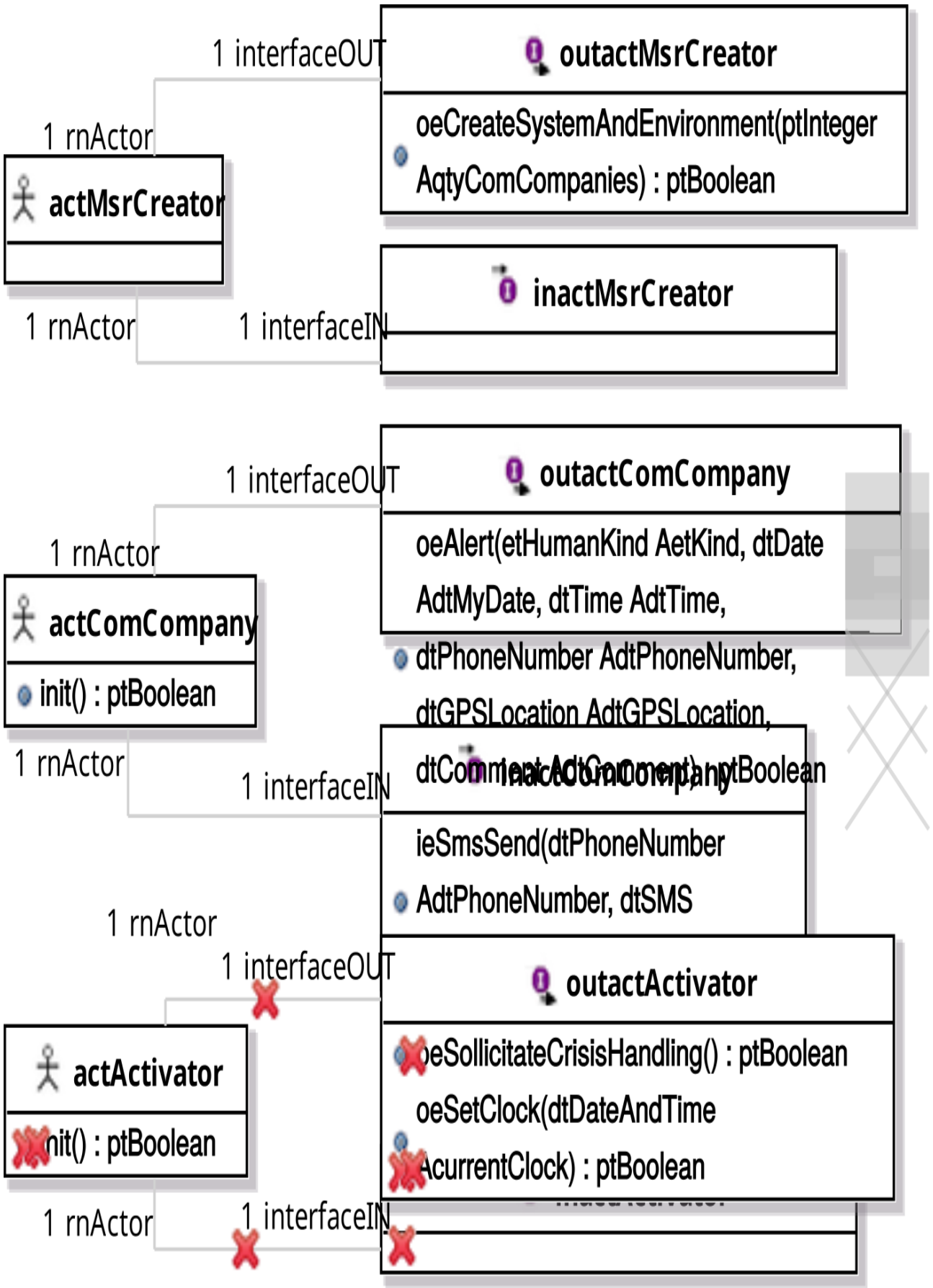


Fig. 3.2 Environment Model - Local View 02. Part 2.

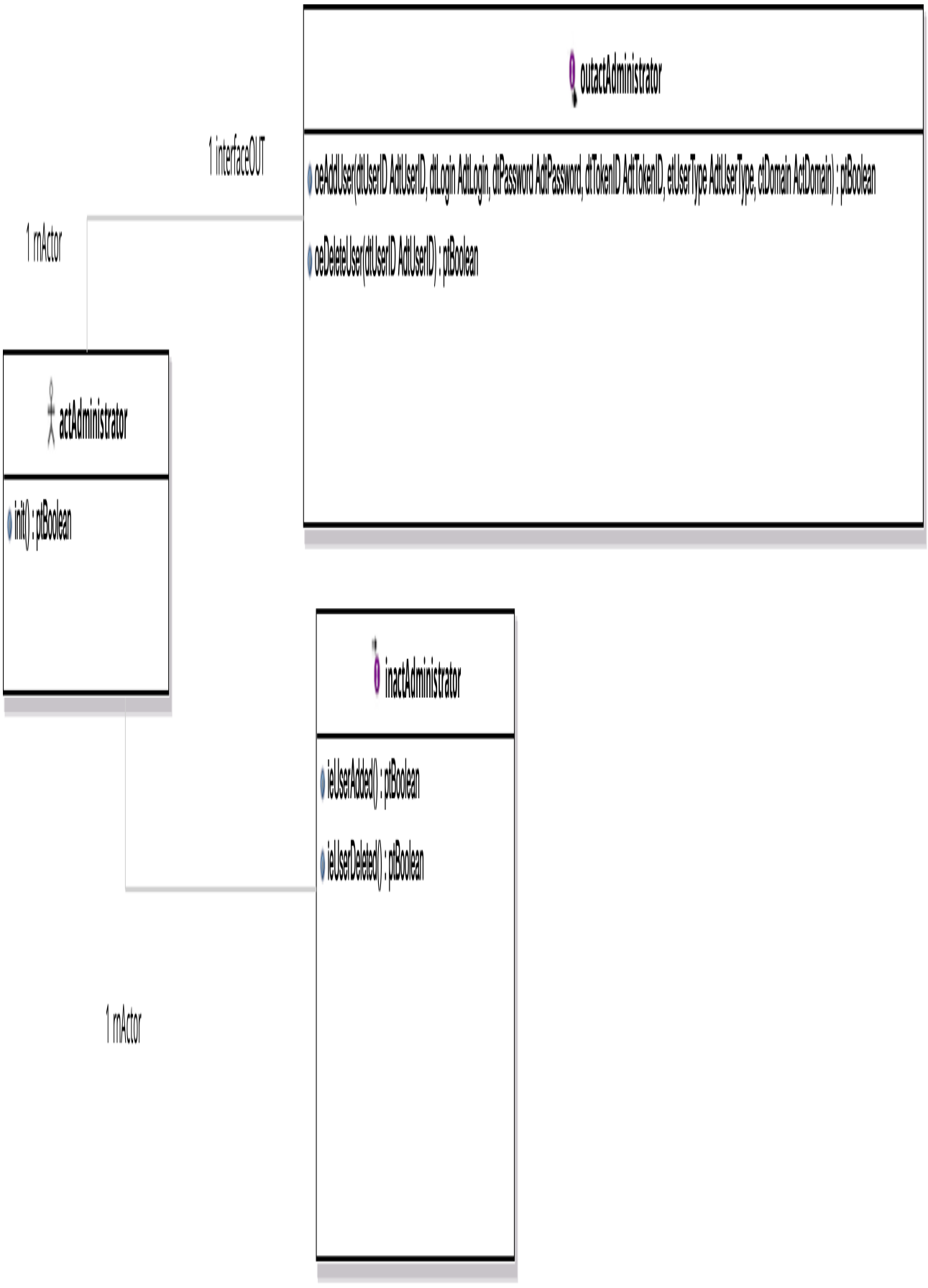


Fig. 3.3 Environment Model - Local View 03. .

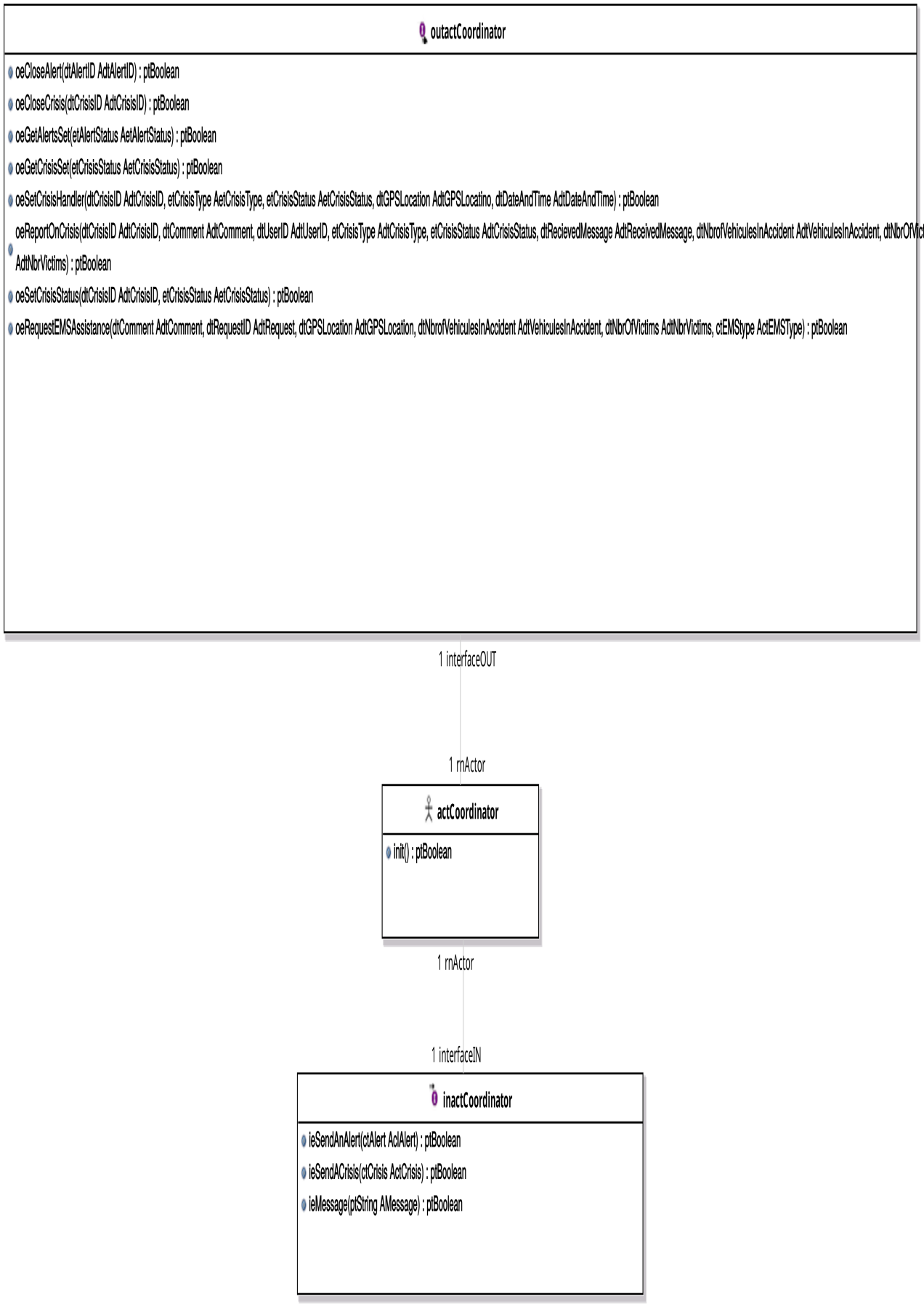


Fig. 3.4 Environment Model - Local View 04. .

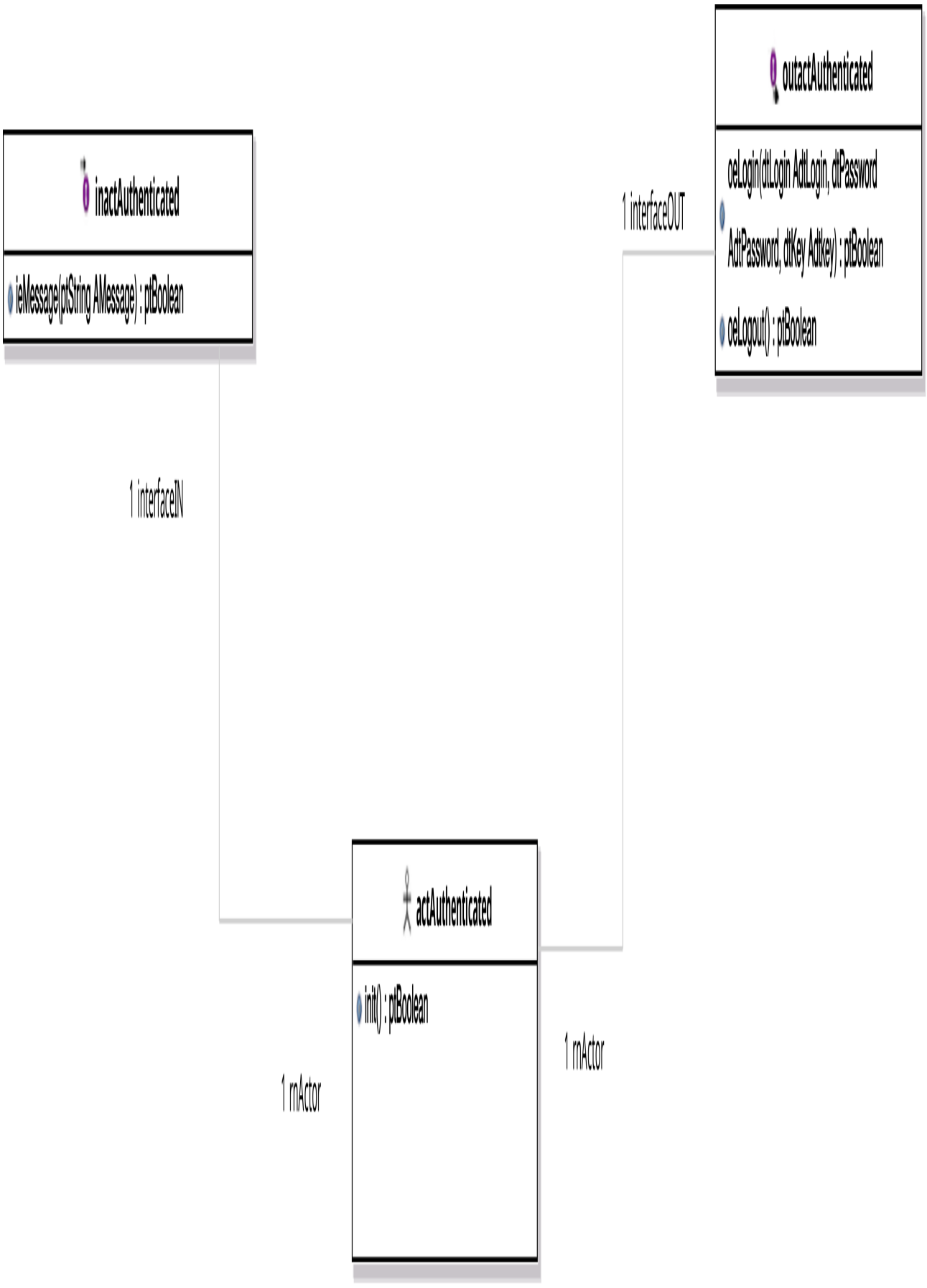


Fig. 3.5 Environment Model - Local View 05. .

3.6 Local view 07

Figure 3.6

3.7 Actors and Interfaces Descriptions

We provide for the given views the description of the actors together with their associated input and output interface descriptions.

3.7.1 ***actDomainExpert*** *Actor*

Actor	
actDomainExpert	
InputInterfaces	
IN 1	**ieAlertRecieved():ptBoolean**
OutputInterfaces	
OUT 1	**oeValidateAlert(dtAlertID AdtAlertID,dtGPSLocation AdtGPSLocation,dtDateAndTime AdtDateandTime,etAlertStatus edtAlertStatus,ctDomain ActDomain):ptBoolean**
OUT 2	**oeSollicitateCrisisHandling():ptBoolean**

3.7.2 ***actComCompany*** *Actor*

Actor	
actComCompany A Communication Company works as communication link between witness/anonymous/victim and iCrash message exchange and victim's family and iCrash system. Communication Company's objectives are: - Delivering SMS messages from humans to iCrash phone number - Transmitting SMS messages from iCrash system to Humans (including witness, anonymous, victim, family member) with an SMS enabling device with phone number. The responsibilities of a Communication Company therefore are: - Guaranteeing integrity, confidentiality, reliabiliy and availability of the information and message delivery system for any SMS communication between iCrash and Humans.	
InputInterfaces	
IN 1	**ieSmsSend(dtPhoneNumber AdtPhoneNumber,dtSMS AdtSMS):ptBoolean**
OutputInterfaces	
OUT 1	**oeAlert(etHumanKind AetKind,dtDate AdtMyDate,dtTime AdtTime,dtPhoneNumber AdtPhoneNumber,dtGPSLocation AdtGPSLocation,dtComment AdtComment):ptBoolean**

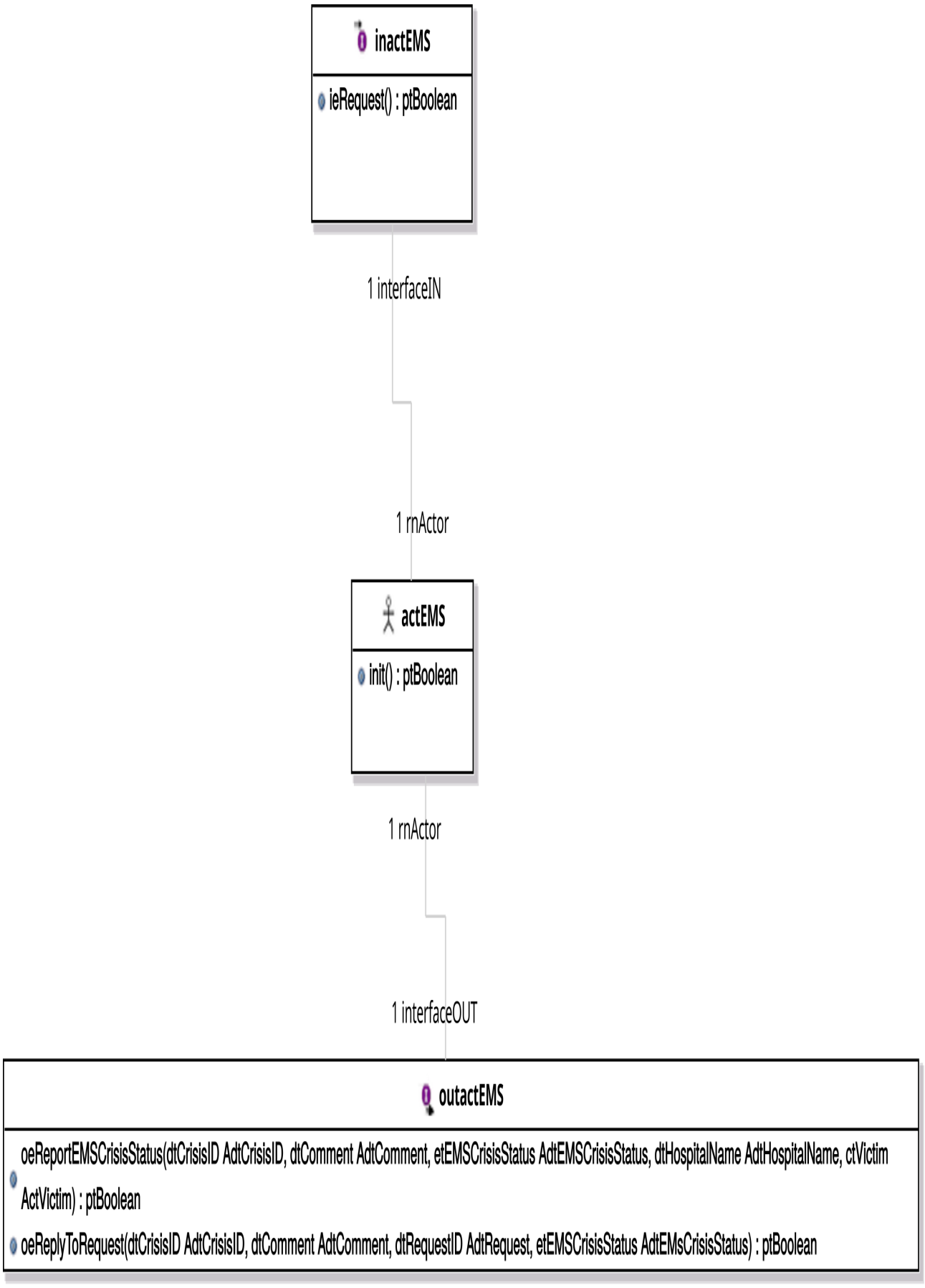


Fig. 3.6 Environment Model - Local View 07. .

3.7.3 actEMS Actor

Actor
actEMS
EMS services is an acronym of Emergency Services. The person operating at EMS end of the system is a human provided with an interface including a login, password and tokenID. The objectives of EMS services are: - To communicate with a Coordinator at company ABC by receiving and sending messages through iCrash - Report back to iCrash when a rescue unit (and its consistency) has been dispatched to the accident site. - Send infromation about the accident back to iCrash Therefore the responsibilities are: - To be aware how to securely use the iCrash system by posessing information about identification. - Provive the information needed to iCrash in order to send the SMS to correct family number
OutputInterfaces
OUT 1 **oeReplyToRequest(dtCrisisID AdtCrisisID,dtComment AdtComment,dtRequestID AdtRequest,etEMSCrisisStatus AdtEMsCrisisStatus, dtHospitalName AdtHospitalName,dtName AdtVictimsName,dtName AdtVictimICEContact-Namme,dtPhoneNumber AdtVictimICEContactPhoneNumber):ptBoolean** OUT 2 **oeReportEMSCrisisStatus(dtCrisisID AdtCrisisID,dtComment AdtComment,etEMSCrisisStatus AdtEMSCrisisStatus):ptBoolean**

3.7.4 actMsrCreator Actor

Actor
actMsrCreator
A Creator is a human working at/for company ABC, possessing high enough technical capabilities to perform the installation of iCrash. In addition to installation of iCrash, a Creator's role also includes definition of values for intial system's state and environment. A Creator and Administrator are separate humans and one cannot behold the tasks of both.
OutputInterfaces
OUT 1 **oeCreateSystemAndEnvironment(ptInteger AqtyComCompanies):ptBoolean**

3.7.5 actCoordinator Actor

Actor
actCoordinator
A Coordinator is a human who has the following objectives: - Monitoring excisting alerts and crisis and to - Managing alerts and crisis until they have been declared resolved To achieve the objectives, the responsibilities for a Coordinator therefore are: - Communicating with EMS though iCrash system in order to request for dispatch of vehicles - To be available for crising handling (requests and alerts) - Being able to make a decision if an alert or or crisis can be closed - To be able to use the system safely regarding identification - To send a message to the victim's family through ComCompany containing carefully chosen information about the accident and its victims.
Extends
icrash.environment.actAuthenticated
InputInterfaces
IN 1 **ieSendAnAlert(ctAlert AclAlert):ptBoolean**

continues in next page . . .

. . . Actor table continuation

IN 2	**ieSendACrisis(ctCrisis ActCrisis):ptBoolean**
IN 3	**ieMessage(ptString AMessage):ptBoolean**
OutputInterfaces	
OUT 1	**oeCloseAlert(dtAlertID AdtAlertID):ptBoolean**
OUT 2	**oeCloseCrisis(dtCrisisID AdtCrisisID):ptBoolean**
OUT 3	**oeGetAlertsSet(etAlertStatus AetAlertStatus):ptBoolean**
OUT 4	**oeGetCrisisSet(etCrisisStatus AetCrisisStatus):ptBoolean**
OUT 5	**oeSetCrisisHandler(etCrisisStatus AetCrisisID):ptBoolean**
OUT 6	**oeReportOnCrisis(dtAlertID AdtCrisisID,dtComment AdtComment,dtUserID AdtUserID,etCrisisType AdtCrisisType, etCrisisStatus AdtCrisisStatus, dtRecievedMessage AdtReceivedMessage, dtNbrofVehiculesInAccident AdtVehiculesInAccident, dtNbrOfVictims AdtNbrVictims):ptBoolean**
OUT 7	**oeSetCrisisStatus(dtCrisisID AdtCrisisID,etCrisisStatus AetCrisisStatus):ptBoolean**
OUT 8	**oeValidateAlert(dtAlertID AdtAlertID):ptBoolean**
OUT 9	**oeInfoFam(dtComment AdtComment,dtPhoneNumber AdtVictimICEContactPhoneNumber,dtHospitalName AdtHospitalName):ptBoolean**
OUT 10	**oeRequestEMSAssistance(dtComment AdtComment,dtRequestID AdtRequest,dtGPSLocation AdtGPSLocation,dtNbrofVehiculesInAccident AdtVehiculesInAccident, dtNbrOfVictims AdtNbrVictims):ptBoolean**

3.7.6 **actAdministrator** *Actor*

ACTOR
actAdministrator
Administrating the iCrash system is the responsibility of an Administrator, who is also a human. To fullfill his/her objective: - Adding users (Coordinators, EMS or DomainExpert) to and from the system and its environment; - Deleting users to and from the system and its environment. The responsibilities include: - Identifying the company ABC employees that are users with access to the system; - Be aware of the security policies of company ABC; - Be aware of the the system's identification information for secure system usage; - Communication with users (Coordinators, EMS or DomainExpert) in order to provide them with identification information to guarantee the secure use of the system.
Extends
icrash.environment.actAuthenticated
InputInterfaces
IN 1 **ieUserAdded():ptBoolean**
IN 2 **ieUserDeleted():ptBoolean**

continues in next page . . .

. . . Actor table continuation

OutputInterfaces	
OUT 1	**oeAddUser(dtUserID AdtUserID, dtLogin AdtLogin, dtPassword AdtPassword, dtTokenID AdtTokenID):ptBoolean**
OUT 2	**oeDeleteUser(dtUserID AdtUserID):ptBoolean**

3.7.7 ***actAuthenticated*** *Actor*

ACTOR
actAuthenticated
Authenticated is a super user of iCrash System whom all other users inherit from for safe use of the system.
InputInterfaces
IN 1 **ieMessage(ptString AMessage):ptBoolean**
OutputInterfaces
OUT 1 **oeLogin(dtLogin AdtLogin, dtPassword AdtPassword, dtKey AdtKey):ptBoolean**
OUT 2 **oeLogout():ptBoolean**

Chapter 4
Concept Model

4.1 PrimaryTypes-Classes

4.1.1 Local view 01

Figure 4.1 Shows the primary types class types.

4.1.2 Local view 02

Figure 4.2

4.1.3 Local view 03

Figure 4.3

4.1.4 Local view 04

Figure 4.4

4.1.5 Local view 06

Figure 4.5

4.1.6 Local view 08

Figure 4.6

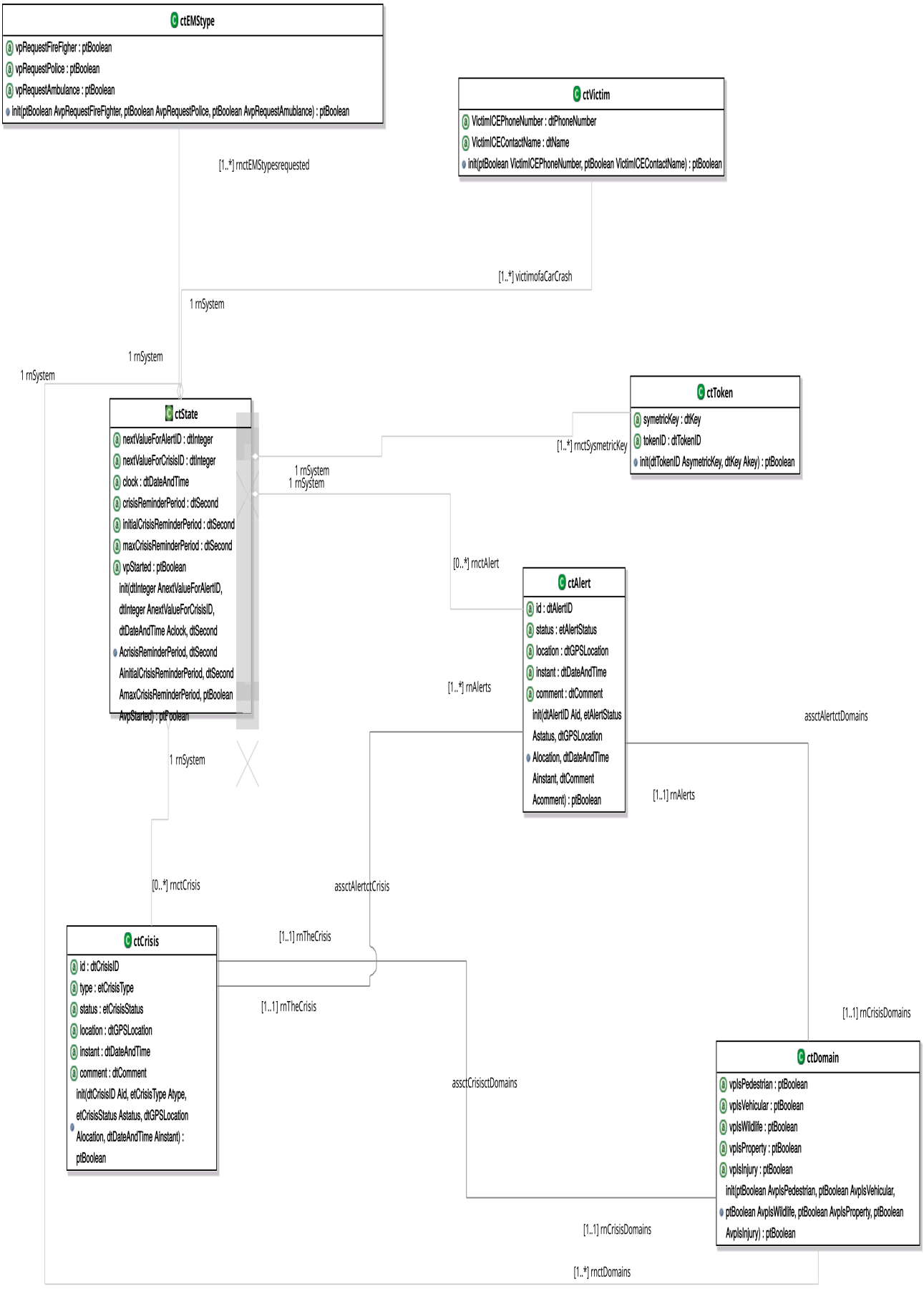


Fig. 4.1 Concept Model - PrimaryTypes-Classes local view 01. Local view of the Primary Types Class Types.

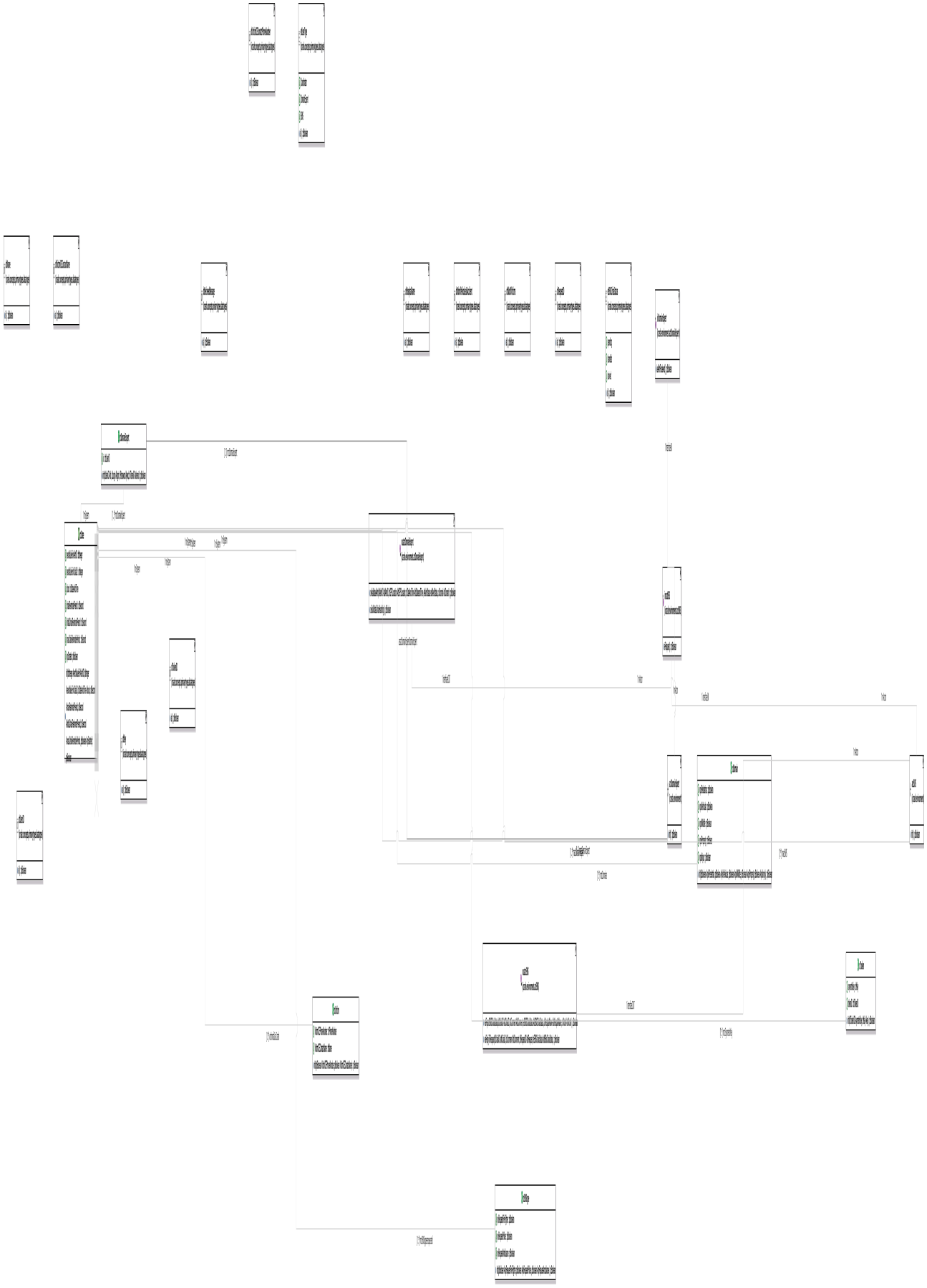


Fig. 4.2 Concept Model - PrimaryTypes-Classes local view 02. .

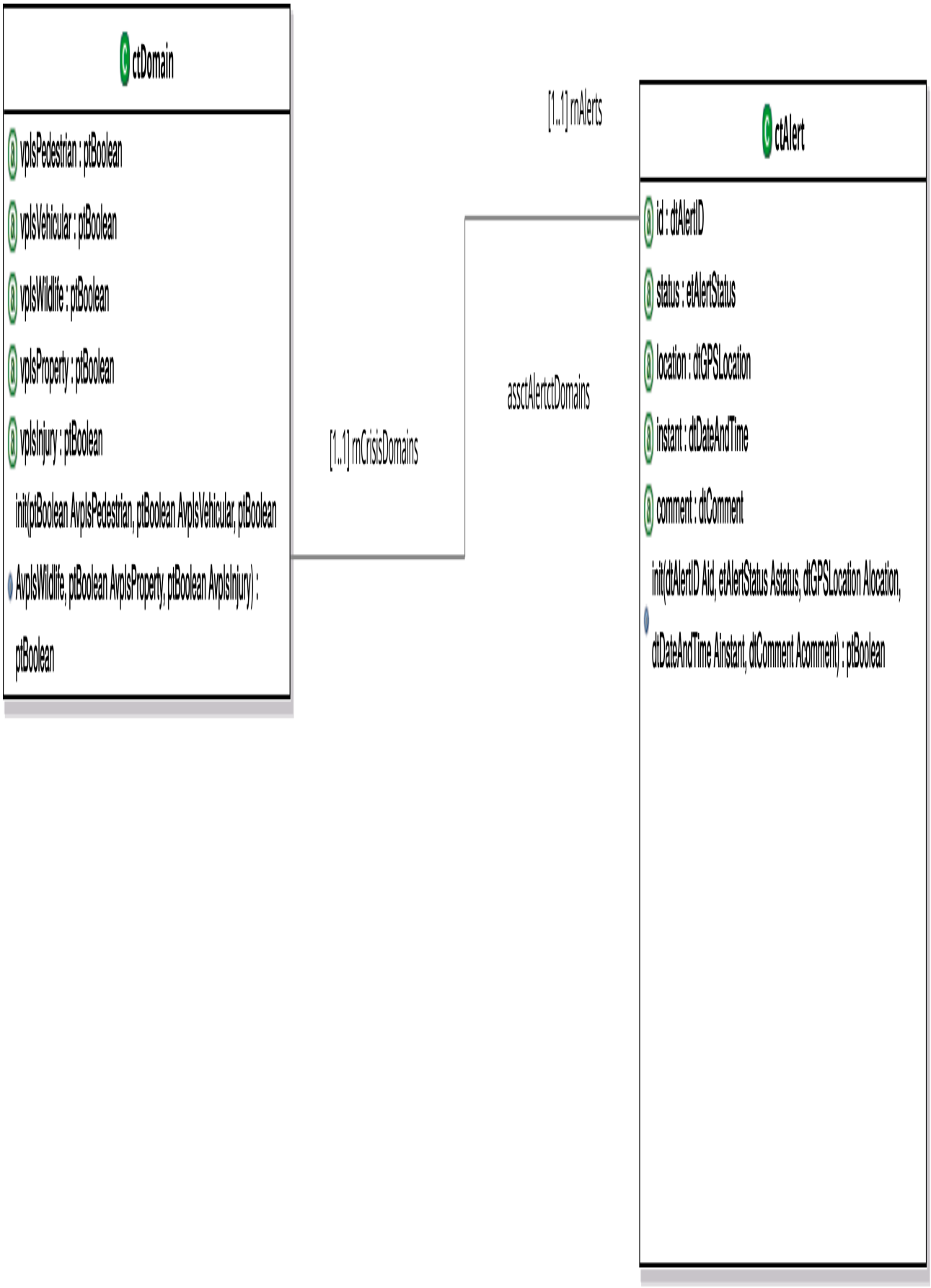


Fig. 4.3 Concept Model - PrimaryTypes-Classes local view 03. .

Fig. 4.4 Concept Model - PrimaryTypes-Classes local view 04. .

Fig. 4.5 Concept Model - PrimaryTypes-Classes local view 06. .

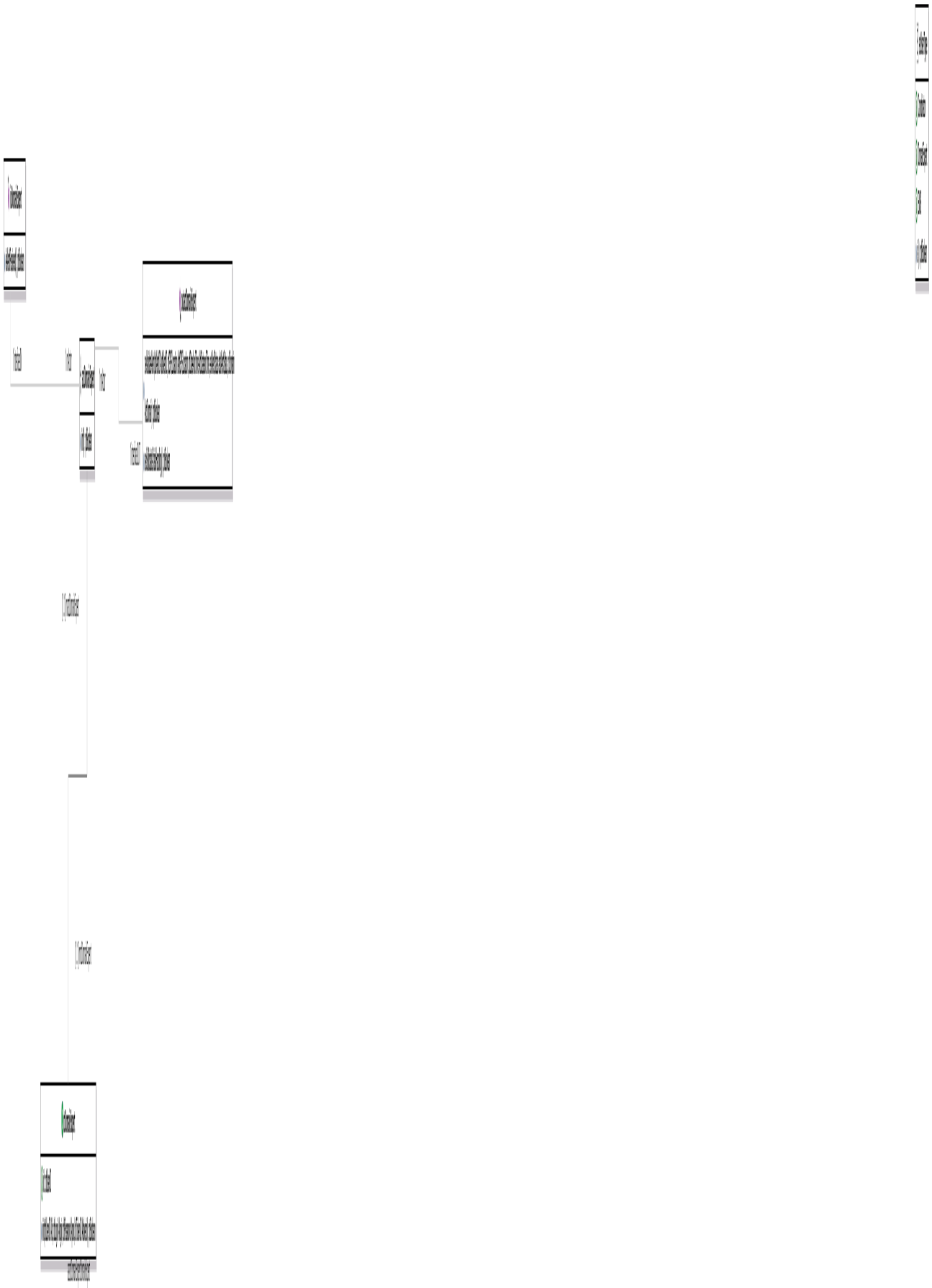


Fig. 4.6 Concept Model - PrimaryTypes-Classes local view 08. .

4.1.7 Global view 01

Figure 4.7 Shows the primary types class types together with the association(s) types with the actor classes of the environment model.

4.2 PrimaryTypes-Datatypes

4.2.1 Global view 01

Figure 4.8 Shows the primary types datatype types.

4.3 SecondaryTypes-Datatypes

4.3.1 Local view 01

Figure 4.9 Shows the secondary types datatype types.

4.4 Concept Model Types Descriptions

This section provides the textual descriptions of all the types defined in the concept model and that can be part of the graphical views provided.

4.4.1 Primary types - Class types descriptions

The table below is providing comments on the graphical views given for the class types of the primary types. Type logical operations are precisely specified in the operation model.

Classes	
ctDomainExpert	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
attribute	`id: dtUserID`:
operation	`init`:
ctCrisis	
This class is a crisis which is initiated from the alerts received by the system.	
attribute	`id`: this attribute is an ID of the crisis. The type is a string.
attribute	`type`: this attribute is a status of a crisis. The status is enum and it could take as a value: ”small“, ”medium“, ”huge“.
attribute	`status`: this attribute is a type of a crisis. The type is enum and it could take as a value: ”pending“, ”handled“, ”solved“.
attribute	`location`: this attribute is a GPS location of a crisis. The type is dtGPSLocation which contains real numbers for latitude and longitude.

continues in next page . . .

. . . Classes table continuation

attribute	`instant`: this attribute is a date and a time when crisis happens.
attribute	`comment`: this attribute is a miscellaneous text sent as a comment.
operation	`init`: this operation initializes ctCrisis with necessary attributes.
ctToken	
This class contains information related to a token. Token is used upon logging in in order to provide symmetric encryption and enhanced security.	
attribute	`symetricKey: dtKey`:
attribute	`tokenID: dtTokenID`:
operation	`init`:
ctHuman	
This class is a human that sends an alert to the system through the communication company.	
attribute	`id`: this attribute is an ID of a human. The type is a string which covers the phone numbers of human who reports an alert of a crisis.
attribute	`kind`: this attribute is a status of a human. The status is enum and it could take as a value: ”witness“, ”victim“, ”anonymous“.
operation	`init`: this operation initializes ctHuman with necessary attributes.
ctCoordinator	
This class is a main user of the system. It inherits from the ctAuthenticated the login information (login name and password).	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
attribute	`id`: this attribute is an ID of the coordinator user. The type is a string.
operation	`init`: this operation initializes ctCoordinator with necessary attributes.
ctState	
This class is the main class that represents the system and the state of the system.	
attribute	`nextValueForAlertID`: this attribute keeps track of the next value for an ID of an alert.
attribute	`nextValueForCrisisID`: this attribute keeps track of the next value for an ID of a crisis.
attribute	`clock`: this attribute keeps the track of time.
attribute	`crisisReminderPeriod`:
attribute	`initialCrisisReminderPeriod`:
attribute	`maxCrisisReminderPeriod`:
attribute	`vpStarted`:
operation	`init`: this operation initializes ctState with necessary attributes.
ctAdministrator	
This class is in charge of administrating the system and adding/removing coordinator users.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
operation	`init`: this operation initializes ctAdministrator with necessary attributes.
ctAlert	
This class is an alert sent by the Communication Company from the Human to the system and is assigned to a crisis.	
attribute	`id`: this attribute is an ID of an alert. The type is string.
attribute	`status`: this attribute is a status of an alert. The type is enum and it could take as a value: ”pending“, ”valid“, ”invalid“.
attribute	`location`: this attribute is a GPS location of an alert. The type is dtGPSLocation which contains real numbers for latitude and longitude.
attribute	`instant`: this attribute is a date and a time when alert is received.
attribute	`comment`: this attribute is a miscellaneous text sent as a comment.
operation	`init`: this operation initializes ctAlert with necessary attributes.
ctAuthenticated	
This abstract class contains the login information of the system users.	
attribute	`login`: this attribute is a login name of the system user. The type is a string.
attribute	`pwd`: this attribute is a lpassword of the system user. The type is a string.
attribute	`vpIsLogged`: this boolean attribute takes into account if the user is logged or not.
operation	`init`: this operation initializes ctAuthenticated with necessary attributes.
ctEMStype	

continues in next page . . .

. . . Classes table continuation

attribute	`vpRequestFireFigher: ptBoolean:`
attribute	`vpRequestPolice: ptBoolean:`
attribute	`vpRequestAmbulance: ptBoolean:`
operation	`init:`
ctVictim	
extends	icrash.concepts.primarytypes.classes.ctHuman
attribute	`VictimICEPhoneNumber: dtPhoneNumber:`
attribute	`VictimICEContactName: dtName:`
operation	`init:`
ctDomain	
This class contains information regarding Accident domains, every alert gets assigned one. This helps to categorize accidents and to provide best possible Coordinator with specified knowledge to each crisis. The accident domain is assigned by a Domain Expert.	
attribute	`vpIsPedestrian: ptBoolean:`
attribute	`vpIsVehicular: ptBoolean:`
attribute	`vpIsWildlife: ptBoolean:`
attribute	`vpIsProperty: ptBoolean:`
attribute	`vpIsInjury: ptBoolean:`
operation	`init:`

4.4.2 Primary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the primary types.

Datatypes	
dtHumanName	
This presents the name of the accident victim or for example a witness, whatever the system Coordinator thinks is important to save into the system regarding the crisis.	
extends	null
operation	`is:`
dtRecievedMessage	
This descibes the message/messages received to iCrash.	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtLogin	
Every user (actAdministrator, actCoordinator, actDomainExpert, actEMS) must have a username for logging in.	
extends	null
operation	`is:`
dtAlertID	
Every alert is auto-assigned an AlertID by the system.	
extends	null
operation	`is:`
dtKey	
Unique key generated by a token (with a unique tokenID) for users to log into the system using symmetric encryption.	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtPassword	

continues in next page . . .

. . . Datatypes table continuation

Every user with a login and token must also have a password. Password is entered upon a login operation.	
extends	null
operation	`is:`
dtLongitude	
Describes the longitude GPS-coordinates of the accident venue.	
extends	null
operation	`is:`
dtNbrOfVictims	
extends	lu.uni.lassy.messir.libraries.math.dtReal
operation	`is:`
dtRequestID	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtPhoneNumber	
This datatype includes the phone number of for example witness who sent a message about the accident or phone number of the vitim's family.	
extends	null
operation	`is:`
dtNbrofVehiculesInAccident	
extends	lu.uni.lassy.messir.libraries.math.dtReal
operation	`is:`
dtVictimICEContactName	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtGPSLocation	
Describes the longitude and latitude GPS-coordinates of the accident location.	
attribute	`latitude: dtLatitude:`
attribute	`longitude: dtLongitude:`
operation	`is:`
operation	`isNearTo:`
dtName	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtComment	
This describes a comment that can be inserted into an accident report or a message.	
extends	null
operation	`is:`
dtUserID	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	`is:`
dtLatitude	
Describes the latitude GPS-coordinates for the accident location.	
extends	null
operation	`is:`
dtCrisisID	
Every crisis gets auto-assigned an ID.	
extends	null

continues in next page . . .

... Datatypes table continuation

operation	is:
dtTokenID	
Every token has a unique tokenID, therefore every human with a login has a unique tokenID.	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	is:
dtHospitalName	
This describes the name of the possible hospital where the victim's were taken to. This same hospital name is sent to the victim's family for later reunion.	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	is:
dtVictimICEContactPhoneNumber	
extends	lu.uni.lassy.messir.libraries.string.dtString
operation	is:

Enumerations	
etHumanKind	
Describes the sender of the original message (victim, witness, anonymous).	
operation	is:
etEMSCrisisStatus	
operation	is:
etUserType	
operation	is:
etCrisisType	
Describes the type of the crises (Pedestrian, Wildlife, Property, Injury).	
operation	is:
etCrisisStatus	
Describes the status of the crisis (pending, active, resolved).	
operation	is:
etAlertStatus	
Describes the status of the alert (valid, invalid).	
operation	is:

4.4.3 Primary types - Association types descriptions

The table below is providing comments on the association types of the primary types.

Undirected associations
assctDomainExpertDomainExpert
assctHumanactComCompany
The initial SMS sent by witness/victim/anonymous gets delivered to iCrash by Communication Company.
assctAlertctCrisis
Every alert is turned into a crisis after the alert has been validated by the Coordinator.
assctAlertctHuman
Every alert is associated with a human (witness, anonymous, victim) who sent the initial SMS about the accident to iCrash.

continues in next page ...

... Undirected associations table continuation

assctCrisisctCoordinator Every crisis has as specialist handling it, this specialist is called a Coordinator.
assctCrisisctDomains Every crisis gets assigned a Domain (pedestrian, wildlife, property, injury).
assctAlertctDomains Every crisis must be assigned a domain (Wildlide, Pedestrian, Property, Injury).
assctAuthenticatedctToken Every authenticated user must have a token.
assctAuthenticatedactAuthenticated
assctCoordinatoractCoordinator
assctCoordinatorctDomains Every coordinator is assigned an alert linked to a specific accident domain by the Domain Expert.

4.4.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

4.4.5 Primary types - Composition types descriptions

There are no composition types for the primary types.

4.4.6 Secondary types - Class types descriptions

There are no elements in this category in the system analysed.

4.4.7 Secondary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the secondary types.

DATATYPES	
dtSMS This describes the message containing detailed information about the accident and its victimes sent to the family.	
attribute	`value: null:`
operation	`is:`

4.4.8 Secondary types - Association types descriptions

There are no association types for the secondary types.

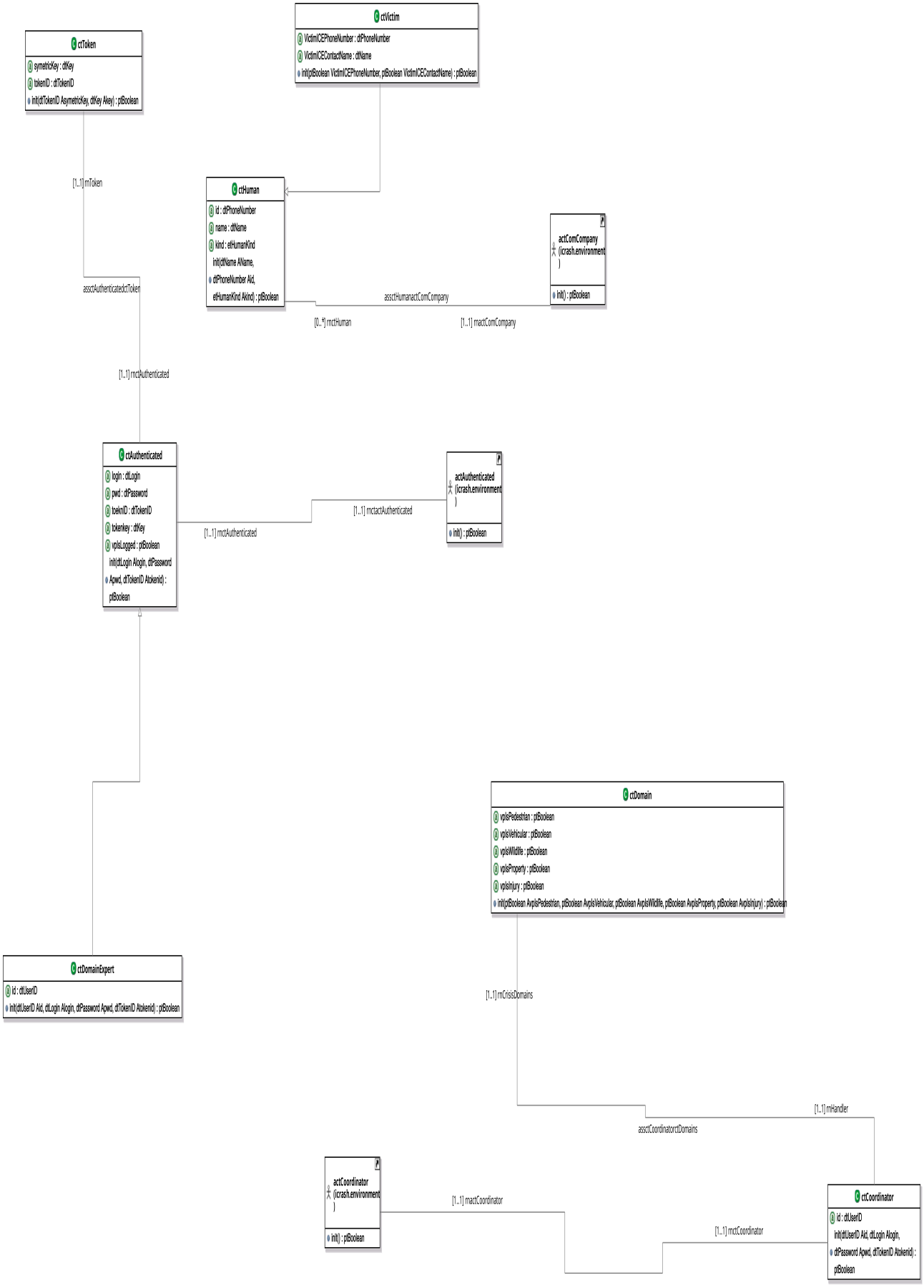


Fig. 4.7 Concept Model - PrimaryTypes-Classes global view 01. Associations between primary type classes and Actors..

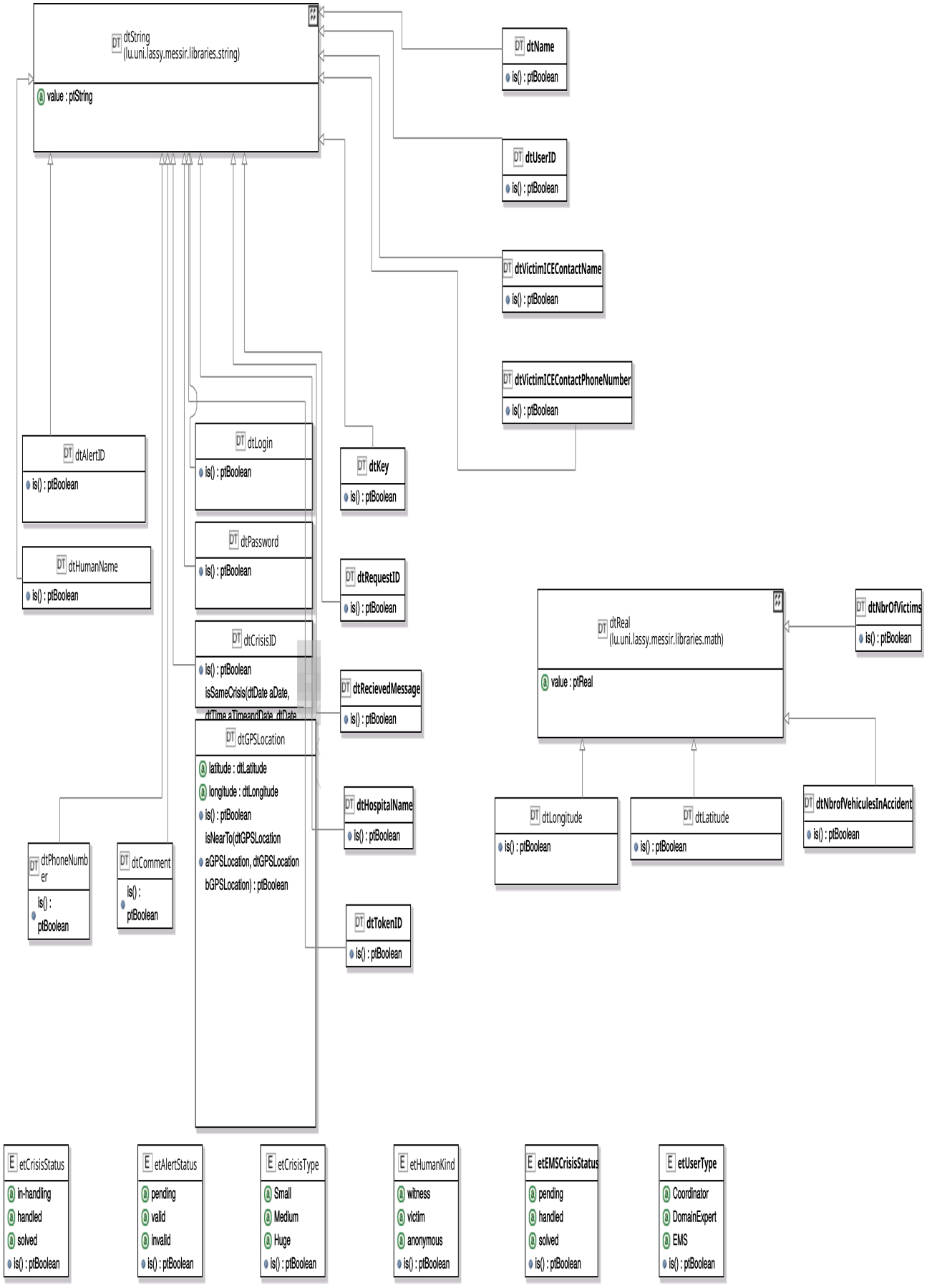


Fig. 4.8 Concept Model - PrimaryTypes-Datatypes global view 01. .

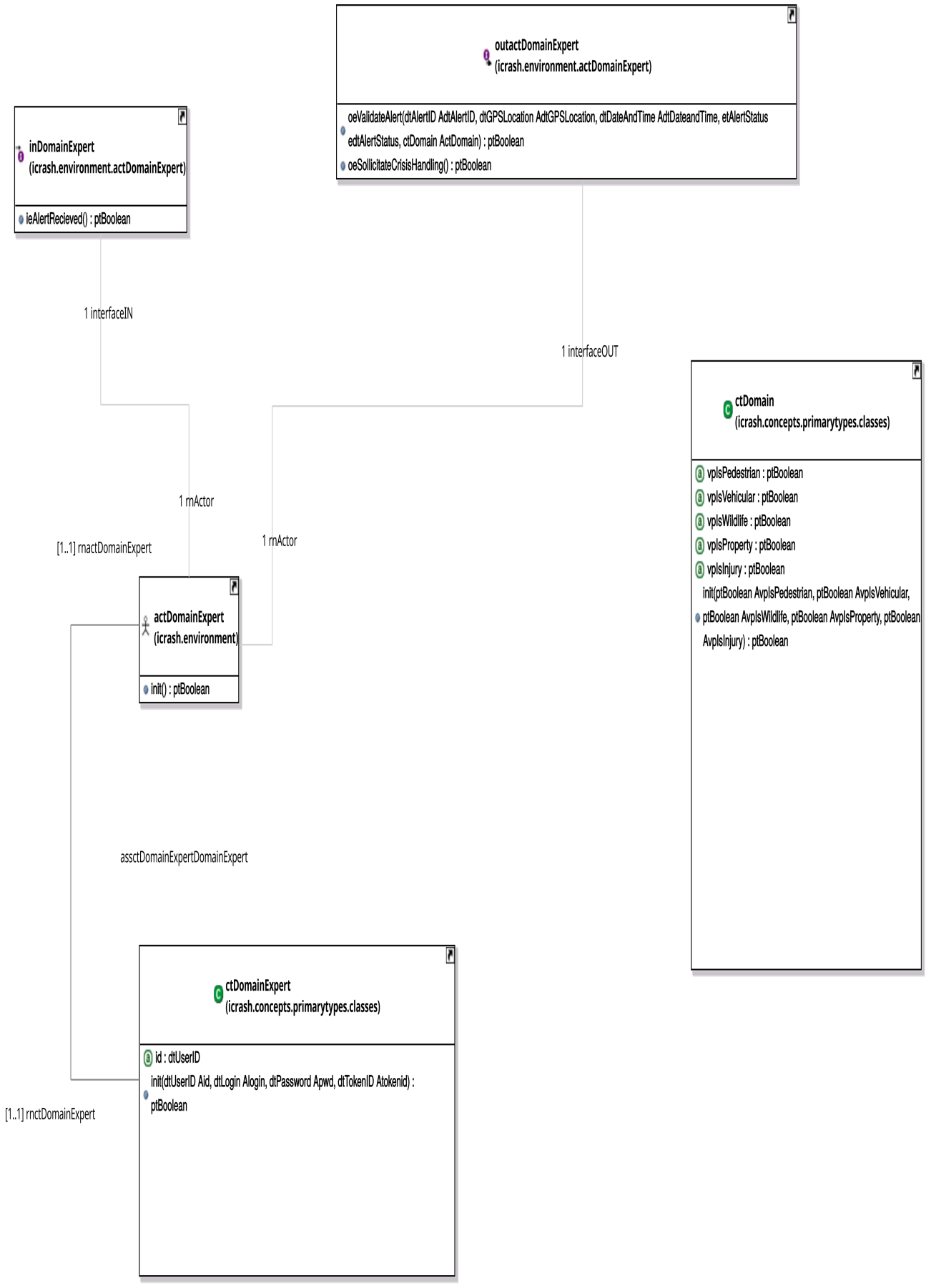


Fig. 4.9 Concept Model - SecondaryTypes-Datatypes local view 01. Local view of the secondary types datatype types.

4.4.9 Secondary types - Aggregation types descriptions

There are no aggregation types for the secondary types.

4.4.10 Secondary types - Composition types descriptions

There are no composition types for the secondary types.

Chapter 5
Operation Model

This section contains the operation schemes of each operation defined in either an actor, its output interface, or in a primary or secondary type (class, datatype or enumeration types). The Messir OCL code listing is joined to the comment table.

5.1 Environment - Out Interface Operation Scheme for actAdministrator

5.1.1 Operation Model for oeAddUser

The `oeAddUser` operation has the following properties:

Operation	
oeAddUser	
ActAdministrator's goal is to add a new Coordinator to the system as a CoordinatorActor, so the the Coordinator can safely use the system	
Parameters	
1	**dtUserID** AdtUserID
2	**dtLogin** AdtLogin
3	**dtPassword** AdtPassword
4	**dtTokenID** AdtTokenID
5	**etUserType** AdtUserType
6	**ctDomain** ActDomain
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	No Coordinator can have the same Token
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. the ctAdministartorActor has is logged in.
Post-Condition (functional)	
PostF 1	A new Coordinator has been aded to the system as a CoordinatorActor and he is now able to use the sytem safely.
Post-Condition (protocol)	

continues in next page . . .

... Operation table continuation

PostP 1	1. A new Coordinator has been added to the sytem with his own token, username, password and Domains
Example	
The ctAdministratorActor actor bill sends the message oeAddCoodrinator(1,Steve, pwdMessirExcalibur2017, 12345, 'Vehicular' 'Pedestrian') to setup a crisis management team made one coordinator and indicating his identification information in terms of ID (i.e. 1), password (i.e.pwdMessirExcalibur2017), tokenID (i.e. 12345) and Domain (i.e.'Vehicular"Pedestrian')	

The listing 5.1 provides the Messir OCL description of the operation.

Listing 5.1 Messir OCL description of the operation *oeAddUser*.

5.1.2 Operation Model for oeDeleteUser

The `oeDeleteUser` operation has the following properties:

Operation	
oeDeleteUser	
A Coordinator is beeing deleted from the system by an Administrator.	
Parameters	
1	**dtUserID** AdtUserID
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. the Coordinator has previously been added to the system.
Post-Condition (functional)	
PostF 1	1. the Coordinator is removed from the system no longer enabling him to use or access the system safely.
Post-Condition (protocol)	
PostP 1	1. All data concerning the Coordinator not including his crisises is removed from the system.
Example	
the ctAdministartorActor bill sends the message oeDeleteCoordinator(1) deleting the Coordinator with the ID (i.e. 1) from the system.	

The listing 5.2 provides the Messir OCL description of the operation.

Listing 5.2 Messir OCL description of the operation *oeDeleteUser*.

5.2 Environment - Out Interface Operation Scheme for actAuthenticated

5.2.1 Operation Model for oeLogin

The `oeLogin` operation has the following properties:

Operation	
oeLogin	
An authenticated user wants to login to the system to use it safely, this is done by entering a username, password and a key generated by a token.	
Parameters	
1	**null** AdtLogin
2	**null** AdtPassword
3	**null** AdtKey
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. the authenticated user that is trying to login has to exsist within the system.
Post-Condition (functional)	
PostF 1	The authenticated user has now been logged into the system and can now use it safely.
Post-Condition (protocol)	
PostP 1	The the protocol variable for the authenticated user executing this opperation, preP must be vpLogged = true
Example	
ctAdministartorActor actor bill sends the message oeLogin(icrashadmin,7WXC1359) to securly connect to the system.	

5.2.2 Operation Model for oeLogout

The `oeLogout` operation has the following properties:

Operation	
oeLogout	
An AuthentcatedActor aims to disconnect from the system.	
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. The AuthenticatedActor has to be logged in.
Post-Condition (functional)	
PostF 1	The AutenticatedActor has been disconnected from the system.
Post-Condition (protocol)	

continues in next page . . .

... Operation table continuation

PostP 1	The the protocol variable for the authenticated user executing this opperation, preP must be vpLogged = false
Example	
the ctAdministratorActor bill sends the message oeLogout() to disconnect from the system.	

5.3 Environment - Out Interface Operation Scheme for actComCompany

5.3.1 Operation Model for oeAlert

The `oeAlert` operation has the following properties:

Operation	
oeAlert	
The ComCompany transfers a decleration of a car crash by an etHumanKind indicating specific information.	
Parameters	
1	**null**
	AetKind
2	**null**
	AdtMyDate
3	**null**
	AdtTime
4	**null**
	AdtPhoneNumber
5	**null**
	AdtGPSLocation
6	**null**
	AdtComment
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. a message has been sent by a etHuman to ComCompany.
Post-Condition (functional)	
PostF 1	A crisis is received by the system.
Post-Condition (protocol)	
PostP 1	A crisis is created.
Example	
the ctComCompanyActor actor tango sends the message oeAlert(witness, 2017-11-26-at-10-10-16AM, +3524666445252, 49.627675-6.159590,'3 cars invloved in an accident.')	

The listing 5.3 provides the Messir OCL description of the operation.

```
1  iyyiyuuouhuhuhulhhu
2  hii
3  preF: if coordinator
```

Listing 5.3 Messir OCL description of the operation *oeAlert*.

5.4 Environment - Out Interface Operation Scheme for actCoordinator

5.4.1 Operation Model for oeCloseAlert

The `oeCloseAlert` operation has the following properties:

Operation	
oeCloseAlert	
A Coordinator closes the alert at the same time as the crisis is closed.	
Parameters	
1	**null** AdtAlertID
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. An alert was received by our system. 3. The alert was handled by a coordinator.
Post-Condition (functional)	
PostF 1	1. The crisis is closed and stored in the database.
Post-Condition (protocol)	
PostP 1	The crisis proctocol variable indicating that a crisis is not closed must change
Example	

The listing 5.4 provides the Messir OCL description of the operation.

Listing 5.4 Messir OCL description of the operation *oeCloseAlert*.

5.4.2 Operation Model for oeCloseCrisis

The `oeCloseCrisis` operation has the following properties:

Operation	
oeCloseCrisis	
ctCoordinatoActor wants to declare a crisis as closed after all the necessary information has been received from EMS.	
Parameters	
1	**null** AdtCrisisID
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	

continues in next page . . .

... Operation table continuation

PreP 1	1. the iCrash system has been deployed 2. An alert was received by our system 3. The alert was handled by a coordinator.
Post-Condition (functional)	
PostF 1	1. The crisis is closed and stored in the database.
Post-Condition (protocol)	
PostP 1	The crisis proctocol variable indicating that a crisis is not closed must change
Example	
the ctCoordinatorActor actor Steve sends teh message oeCloseCrisis(1) to declare the crisis with ID equal to 1 as closed.	

5.4.3 Operation Model for oeGetAlertsSet

The `oeGetAlertsSet` operation has the following properties:

OPERATION	
oeGetAlertsSet	
The actCoordinator -actor wants to reecive information about all the pending alerts.	
Parameters	
1	**null** AetAlertStatus
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed.
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	
Example	

5.4.4 Operation Model for oeGetCrisisSet

The `oeGetCrisisSet` operation has the following properties:

OPERATION	
oeGetCrisisSet	
the ctCoordinatorActor wants to receve information about all the pending crisis	
Parameters	
1	**null** AetCrisisStatus
Return type	
null	
Pre-Condition (functional)	

continues in next page ...

. . . Operation table continuation

PreF 1
Pre-Condition (protocol)
PreP 1 1. the iCrash system has been deployed.
Post-Condition (functional)
PostF 1 The ctCoordinatorActor has now recieved informatio about all the pending crisis
Post-Condition (protocol)
PostP 1
Example
the ctCoordinatorActor actor Steve sends the message oeGetCrisisSet(pending) to recieve information about all the pending crisis

5.4.5 Operation Model for oeSetCrisisHandler

The `oeSetCrisisHandler` operation has the following properties:

OPERATION
oeSetCrisisHandler Actor DomainExpert wants to declare that he is taking care of the crisis.
Parameters
1 **null** AetCrisisID
Return type
null
Pre-Condition (functional)
PreF 1
Pre-Condition (protocol)
PreP 1 1. the iCrash system has been deployed. 2. a Coordinator has been added to the system. 3. an Alert has been recieved by the system and a coordinator with the correct Domain is aware of the crisis.
Post-Condition (functional)
PostF 1 the Coordinator has declared that he is taking care of the crisis
Post-Condition (protocol)
PostP 1 the protocol variable of the crisis has changed
Example
the ctCoordinatorActor actor steve sends the message oeSetCrisisHandler(1) ot declare that he is taking care of the crisis wiht the ID equal to 1.

5.4.6 Operation Model for oeReportOnCrisis

The `oeReportOnCrisis` operation has the following properties:

OPERATION
oeReportOnCrisis ctCoordinatorActor wants to set the report for the crisis.
Parameters
1 **null**

continues in next page . . .

... Operation table continuation

	AdtCrisisID
2	**null**
	AdtComment
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. A crisis has been handled.
Post-Condition (functional)	
PostF 1	A report on the crisis has been added to the system.
Post-Condition (protocol)	
PostP 1	
Example	
the ctCoordinaorActor actor steve sends the message oe(1,'3 victims sent to the hospital, 2 cars evacuated and 4 rescure units mobilized') to set the report for the crisis with an ID equal to 1 which he is handling."	

5.4.7 Operation Model for oeSetCrisisStatus

The `oeSetCrisisStatus` operation has the following properties:

Operation	
oeSetCrisisStatus the ctCorodinatorActor wants to change the declaration of the crisis.	
Parameters	
1	**null** AdtCrisisID
2	**null** AetCrisisStatus
Return type	
null	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	1. the iCrash system has been deployed. 2. a Coordinator has been added to the system. 3. an Alert has been recieved by the system and a coordinator with the correct Domain is aware of the crisis.
Post-Condition (functional)	
PostF 1	the declaration to the crisis has changed.
Post-Condition (protocol)	
PostP 1	the protocol variable of the status of the crisis has changed.
Example	
the ctCoordinator actor steve sends the message oeSetCrisisStatus(1, solved) to declare the crissi with Id eqaul to 1 as solved.	

5.4.8 Operation Model for oeRequestEMSAssistance

The `oeRequestEMSAssistance` operation has the following properties:

Operation	
oeRequestEMSAssistance	
Parameters	
1	**dtComment** AdtComment
2	**dtRequestID** AdtRequest
3	**dtGPSLocation** AdtGPSLocation
4	**dtNbrofVehiculesInAccident** AdtVehiculesInAccident
5	**dtNbrOfVictims** AdtNbrVictims
6	**ctEMStype** ActEMSType
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	
Example	

5.5 Environment - Out Interface Operation Scheme for actDomainExpert

5.5.1 Operation Model for oeValidateAlert

The `oeValidateAlert` operation has the following properties:

Operation	
oeValidateAlert	
Parameters	
1	**dtAlertID** AdtAlertID
2	**dtGPSLocation** AdtGPSLocation
3	**dtDateAndTime** AdtDateandTime
4	**etAlertStatus**

continues in next page . . .

. . . Operation table continuation

	edtAlertStatus
5	**ctDomain**
	ActDomain
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	
Example	

5.5.2 Operation Model for oeSollicitateCrisisHandling

The `oeSollicitateCrisisHandling` operation has the following properties:

OPERATION
oeSollicitateCrisisHandling
Return type
ptBoolean
Pre-Condition (functional)
PreF 1
Pre-Condition (protocol)
PreP 1
Post-Condition (functional)
PostF 1
Post-Condition (protocol)
PostP 1
Example

5.6 Environment - Out Interface Operation Scheme for actEMS

5.6.1 Operation Model for oeReportEMSCrisisStatus

The `oeReportEMSCrisisStatus` operation has the following properties:

OPERATION
oeReportEMSCrisisStatus
Parameters

continues in next page . . .

... Operation table continuation

1	**dtCrisisID**
	AdtCrisisID
2	**dtComment**
	AdtComment
3	**etEMSCrisisStatus**
	AdtEMSCrisisStatus
4	**dtHospitalName**
	AdtHospitalName
5	**ctVictim**
	ActVictim
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	
Example	

5.6.2 Operation Model for oeReplyToRequest

The `oeReplyToRequest` operation has the following properties:

OPERATION	
oeReplyToRequest	
Parameters	
1	**dtCrisisID**
	AdtCrisisID
2	**dtComment**
	AdtComment
3	**dtRequestID**
	AdtRequest
4	**etEMSCrisisStatus**
	AdtEMsCrisisStatus
Return type	
ptBoolean	
Pre-Condition (functional)	
PreF 1	
Pre-Condition (protocol)	
PreP 1	
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	

continues in next page ...

... Operation table continuation

Example

5.7 Environment - Out Interface Operation Scheme for actMsrCreator

5.7.1 Operation Model for oeCreateSystemAndEnvironment

The `oeCreateSystemAndEnvironment` operation has the following properties:

Operation
oeCreateSystemAndEnvironment Actor MsrCreator aims to create the iCrash system and envrioment at company ABC.
Parameters
1 **ptInteger** AqtyComCompanies
Return type
ptBoolean
Pre-Condition (functional)
PreF 1
Pre-Condition (protocol)
PreP 1
Post-Condition (functional)
PostF 1 1. the iCrash system has been deployed.
Post-Condition (protocol)
PostP 1
Example
the ctMSRCreatorActor actor the Creator sends the message oeCreateSystemAndEnviroment(1) requesting the initialization fo the system and its enviroment(made of one adminsistrator identified by clock) and indicating that the number of communication company actor instances for the system's enviroment is 1(identified here by tango)

The listing 5.5 provides the Messir OCL description of the operation.

```
1
2   context Environment::outactMsrCreator::
3   oeCreateSystemAndEnvironment(qtyComCompanies:Integer)
4   /*-------------------------------------------*/
5   pre:
6   /* Pre Protocol:*/
7   /* PreF01 */
8   true
9   /* Pre Functional:*/
10  /* PreP01 */
11  and true
12  /*-------------------------------------------*/
13  post:
14  let TheState:PrimaryTypesClasses::ctState in
15  let AactMsrCreator:Environment::actMsrCreator in
16
17  /* Post Functional:*/
18  /* PostF01 */
19  TheState._init(1,true)
20
21  /* PostF02 */
22  and TheState.rnactComCompany->size() = qtyComCompanies
23  and TheState.rnactComCompany
24    ->forAll(cca:Environment::actComCompany | cca._init())
```

```
25
26   /* PostF03 */
27   and AactMsrCreator._init()
28
29   /* Post Protocol:*/
30   /* PostP01 */
31   and TheState.vpmStarted = true
```

Listing 5.5 Messir OCL description of the operation *oeCreateSystemAndEnvironment*.

5.8 Environment - Actor Operation Scheme for actComCompany

5.8.1 Operation Model for init

The `init` operation has the following properties:

Operation
init
This declares the actor ComCompany.
Return type
null
Pre-Condition (functional)
PreF 1
Post-Condition (functional)
PostF 1
Example

The listing 5.6 provides the Messir OCL description of the operation.

```
1
2   context Environment::actComCompany::_init():Boolean
3   /*-------------------------------------------*/
4   pre: true
5   /*-------------------------------------------*/
6   post:
7   let pIN:Environment::inactComCompany in
8   let pOUT:Environment::outactComCompany in
9   if (
10  /* Post Functional:*/
11  /* Post F01 */
12      self.oclIsNew()
13  /* Post F02 */
14      and pIN.oclIsNew()
15      and pOUT.oclIsNew()
16      and pIN = self.InterfaceIN
17      and pOUT= self.InterfaceOUT
18        )
19      then result = true
20      else result = false
21      endif
22   /*-------------------------------------------*/
```

Listing 5.6 Messir OCL description of the operation *init*.

5.9 Primary Types - Operation Schemes for Class ctAdministrator

5.9.1 Operation Model for init

The `init` operation has the following properties:

Operation	
init	
This operation initialises the class Administrator.	
Parameters	
1	**null**
	Alogin
2	**null**
	Apwd
Return type	
null	
Pre-Condition (functional)	
PreF 1	True
Post-Condition (functional)	
PostF 1	True
Example	

5.10 Primary Types - Operation Schemes for Class ctAlert

5.10.1 Operation Model for init

The `init` operation has the following properties:

Operation	
init	
This operation initialises the class Alert.	
Parameters	
1	**null**
	Aid
2	**null**
	Astatus
3	**null**
	Alocation
4	**null**
	Ainstant
5	**null**
	Acomment
Return type	
null	
Pre-Condition (functional)	
PreF 1	True
Post-Condition (functional)	
PostF 1	True

continues in next page . . .

. . . Operation table continuation

Example

5.11 Primary Types - Operation Schemes for Class ctState

5.11.1 Operation Model for init

The `init` operation has the following properties:

Operation
init This operation initialises the class State.
Parameters
1 **null** AnextValueForAlertID 2 **null** AnextValueForCrisisID 3 **null** Aclock 4 **null** AcrisisReminderPeriod 5 **null** AinitialCrisisReminderPeriod 6 **null** AmaxCrisisReminderPeriod 7 **null** AvpStarted
Return type
null
Pre-Condition (functional)
PreF 1 True
Post-Condition (functional)
PostF 1
Example
True

5.12 Primary Types - Operation Schemes for Datatype dtAlertID

5.12.1 Operation Model for is

The `is` operation has the following properties:

Operation
is Displays the ID for an alert.
Return type
null

continues in next page . . .

... Operation table continuation

Pre-Condition (functional)
PreF 1 True
Post-Condition (functional)
PostF 1 True
Example
The alert ID is:789879

The listing 5.7 provides the Messir OCL description of the operation.

```
1
2   context PrimaryTypesDatatypes::dtAlertID::
3   is():Boolean
4   /*-----------------------------------------*/
5   pre:
6   /* Pre Functional:*/
7   /* Pre F01 */
8   true
9   post:
10  let TheResult:Boolean in
11  if
12  (
13  /* Post Functional:*/
14  /* Post F01 */
15  self.value > 0
16  )
17  then TheResult = true
18  else TheResult = false
19  endif
20  and result = TheResult
21
22  /*-----------------------------------------*/
```

Listing 5.7 Messir OCL description of the operation *is*.

5.13 Primary Types - Operation Schemes for Datatype dtComment

5.13.1 Operation Model for is

The `is` operation has the following properties:

Operation
is A comment can be added or be included in any message received from a human or EMS. Comments can also be used by Coordinators to add comments into reports.
Return type
null
Pre-Condition (functional)
PreF 1 True
Post-Condition (functional)
PostF 1 True
Example
A message from a witness to iCrash: 'Please send ambulance'

The listing 5.8 provides the Messir OCL description of the operation.

```
1
2   context PrimaryTypesDatatypes::dtComment::
3   is():Boolean
4   /*-----------------------------------------*/
5   pre:
6   /* Pre Functional:*/
7   /* Pre F01 */
8   true
9   post:
10  let TheResult:Boolean in
11  let MaxLength:Integer in
12  if
13  (
14  /* Post Functional:*/
15  /* Post F01 */
16  MaxLength = 160
17  and self.value.size() <= MaxLength
18  )
19  then TheResult = true
20  else TheResult = false
21  endif
22  and result = TheResult
23  /*-----------------------------------------*/
```

Listing 5.8 Messir OCL description of the operation *is*.

5.14 Primary Types - Operation Schemes for Datatype dtPhoneNumber

5.14.1 Operation Model for is

The `is` operation has the following properties:

Operation
is   This presents the mobile phone number of the human sending in the initial SMS about the accident as well as the phone number of the family contact.
Return type
null
Pre-Condition (functional)
PreF 1 True
Post-Condition (functional)
PostF 1 True
Example
The phone number of the witness is: 5813935.

The listing 5.9 provides the Messir OCL description of the operation.

```
1
2   context PrimaryTypesDatatypes::dtPhoneNumber::
3   is():Boolean
4   /*-----------------------------------------*/
5   pre:
6   /* Pre Functional:*/
7   /* Pre F01 */
8   true
9   post:
10  let TheResult:Boolean in
11  if
12  (
13  /* Post Functional:*/
14  /* Post F01 */
15  self.value.size() = 10
16  )
17  then TheResult = true
```

```
18   else TheResult = false
19   endif
20   and result = TheResult
21   /*-------------------------------------------*/
```

Listing 5.9 Messir OCL description of the operation *is*.

5.15 Primary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.

5.16 Secondary Types - Operation Schemes for Classes

There are no elements in this category in the system analysed.

5.17 Secondary Types - Operation Schemes for Datatype dtSMS

5.17.1 Operation Model for is

The `is` operation has the following properties:

Operation
is
Return type
null
Pre-Condition (functional)
PreF 1 True
Post-Condition (functional)
PostF 1 True
Example

5.18 Secondary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.

Chapter 6
Test Model(s)

There are no elements in this category in the system analysed.

Chapter 7
Additional Constraints

Appendix A
Use Case Description Tables

A.1 Use cases list(s)

- Summary
  - suDeployAndRun
  - suGlobalCrisisHandling
- user-goal
  - ugAdministrateTheSystem
  - ugManageCrisis
  - ugMonitor
  - ugSecurelyUseSystem
- subfunction
  - oeAddCoordinator
  - oeAlert
  - oeCloseAlert
  - oeCloseCrisis
  - oeCreateSystemAndEnvironment
  - oeDeleteCoordinator
  - oeGetAlertsSet
  - oeSetCrisisHandler
  - oeInfoFam
  - oeLogin
  - oeLogout
  - oeReportOnCrisis
  - oeRequestEMSAssistance
  - oeReportEMSCrisisStatus
  - oeReplyToRequest
  - oeSetCrisisStatus
  - oeSollicitateCrisisHandling
  - oeValidateAlert

- suDeployAndRun
  - oeCreateSystemAndEnvironment
  - ugAdministrateTheSystem
  - suGlobalCrisisHandling
  - oeAlert
- suGlobalCrisisHandling
  - ugSecurelyUseSystem
  - ugMonitor
  - ugManageCrisis
- ugAdministrateTheSystem
  - ugSecurelyUseSystem
  - oeAddUser
  - oeDeleteUser
- ugSecurelyUseSystem
  - oeLogin
  - oeLogout
- ugManageCrisis
  - oeSetCrisisStatus
  - oeSetCrisisHandler
  - oeReportOnCrisis
  - oeRequestEMSAssistance
  - oeCloseCrisis
- ugMonitor
  - oeGetAlertSet
  - oeValidateAlert
  - oeSollicitateCrisisHandling

A.2 Use case Table(s)

A.2.1 Summary

Use-Case Description	
Name	suDeployAndRun
Scope	System
Altitude	Summary
Parameters	
1	none
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actMsrCreator[active]`
2	`actCoordinator[active]`
3	`actComCompany[active]`
Goal(s) description	
The goal is to install the *iCrash* system on its infrastructure and to exploit its capacities related to the secure administration and efficient handling of car crash crisis situations depending on alerts received.	
Reuse	
1	`oeCreateSystemAndEnvironment`
2	`ugAdministrateTheSystem`[1..*]
3	`suGlobalCrisisHandling`[1..*]
4	`oeAlert`[1..*]
Protocol condition(s)	
1	the *iCrash* system has never been deployed and used
Pre-condition(s)	
1	none
Main post-condition(s)	
1	*iCrash* system has been created and able to handled the crisis situations for which it received alerts through the communication company.
Main success steps	
a	the actor `actMsrCreator` executes the `oeCreateSystemAndEnvironment` use case
b	the actor `actAdministrator` executes the `ugAdministrateTheSystem` use case
c	the actor `actComCompany` executes the `oeAlert` use case
d	the actor `actCoordinator` executes the `suGlobalCrisisHandling` use case
Step Constraints Ordering and Extensions	
1	step (a) must be always the first step.
2	step (b) (c) and (d) executions are interleaved.
3	step (b) (c) (d) can be executed multiple times.
4	step (d) can be executed by different `actCoordinator` actors.
5	step (d) can be executed as a reaction to step (c).
Additional Information	
none	

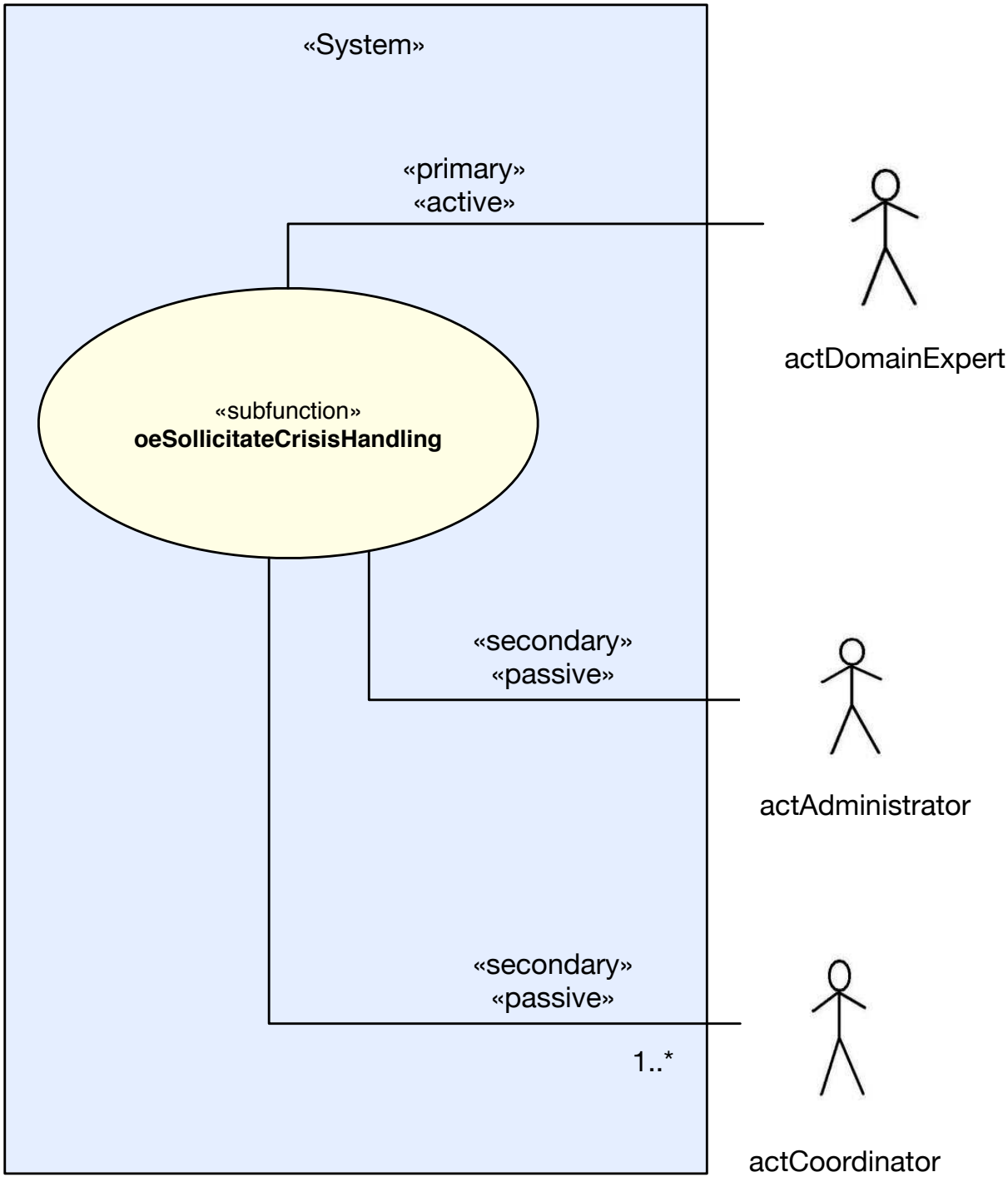


Fig. A.1 *iCrash* Use Case Diagram: oeDeployAndRun

Use-Case Description	
Name	suGlobalCrisisHandling
Scope	System
Altitude	Summary
Parameters	
1	none
Primary actor(s)	
1	`actCoordinator[active]`
2	`actCoordinator[active]`
Secondary actor(s)	
1	none
Goal(s) description	
the `actCoordinator` and `actDomainExpert`'s goals are to monitor the alerts received and the corresponding crises in order to be able to act as necessary to handle the crises.	
Reuse	
1	`ugSecurelyUseSystem`[1..*]
2	`ugMonitor`[1..*]
3	`ugManageCrisis`[1..*]
Protocol condition(s)	
1	the *iCrash* system has been deployed
2	the `actCoordinator` involved in the use case has been declared by the actor `actAdministrator`.
3	the `actDomainExpert` involved in the use case has been declared by the actor `actAdministrator`.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	modifications have been made by the Domain expert on existing alerts.
2	modifications have been made by the Coordinator on the exsisting crises.
Main success steps	
a	the actor `actCoordinator` and the actor `actDomainExpert` execute the `ugSecurelyUseSystem` use case
b	the actor `actDomainExpert` executes the `ugMonitor` use case
c	the actor `actCoordinator` executes the `ugManageCrisis` use case
Step Constraints Ordering and Extensions	
1	steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have there protocol constrained by steps of (a)).
2	step (b) must be executed before (c) can be executed due to domain constrainst related to alerts.
3	steps (a) (b) and (c) can be executed multiple times.
Additional Information	
there might be several actors `actCoordinator` that concurrently execute their `suGlobalCrisisHandling` use cases.	

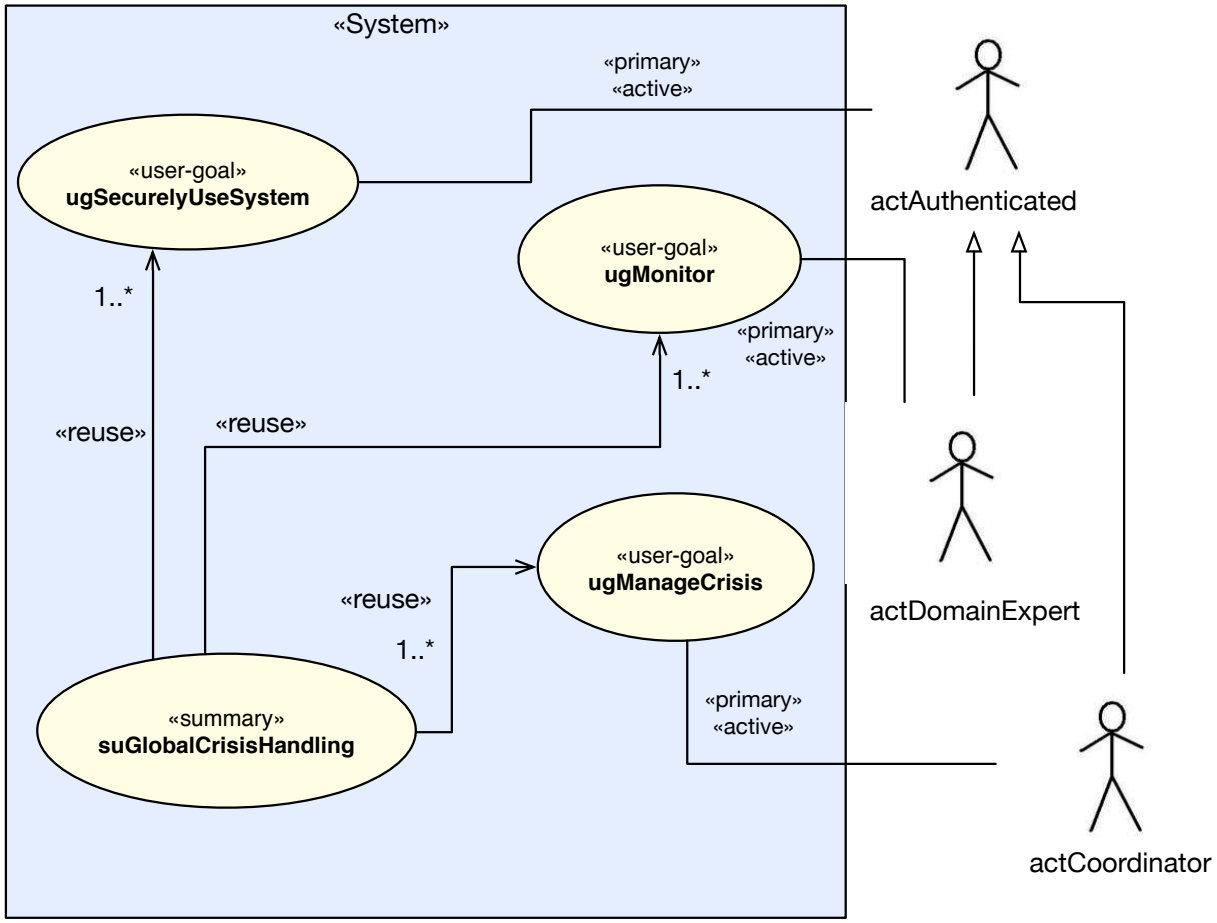


Fig. A.2 *iCrash* Use Case Diagram: ugGlobalCrisisHandling

A.2.2 User-goal

Use-Case Description	
Name	ugAdministrateTheSystem
Scope	System
Altitude	Summary
Parameters	
1	none
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	none
Goal(s) description	
the `actAdministrator`'s goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators that will be granted the responsibility to handle alerts and crisis.	
Reuse	
1	`ugSecurelyUseSystem`[1..*]
2	`oeAddUser`[1..*]
3	`oeDeleteUser`[1..*]
Protocol condition(s)	
1	the *iCrash* system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a new user is added to the system and can now securely use the system.
Main success steps	
a	the actor `actAdministrator` executes the `ugSecurelyUseSystem` use case.
b	the actor `actAdministrator` executes the `oeAddUser` use case.
c	the actor `actAdministrator` executes the `oeDeleteUser` use case.
Step Constraints Ordering and Extensions	
1	steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have there protocol constrained by steps of (a)).
2	steps (a) (b) and (c) can be executed multiple times.
Additional Information	
none	

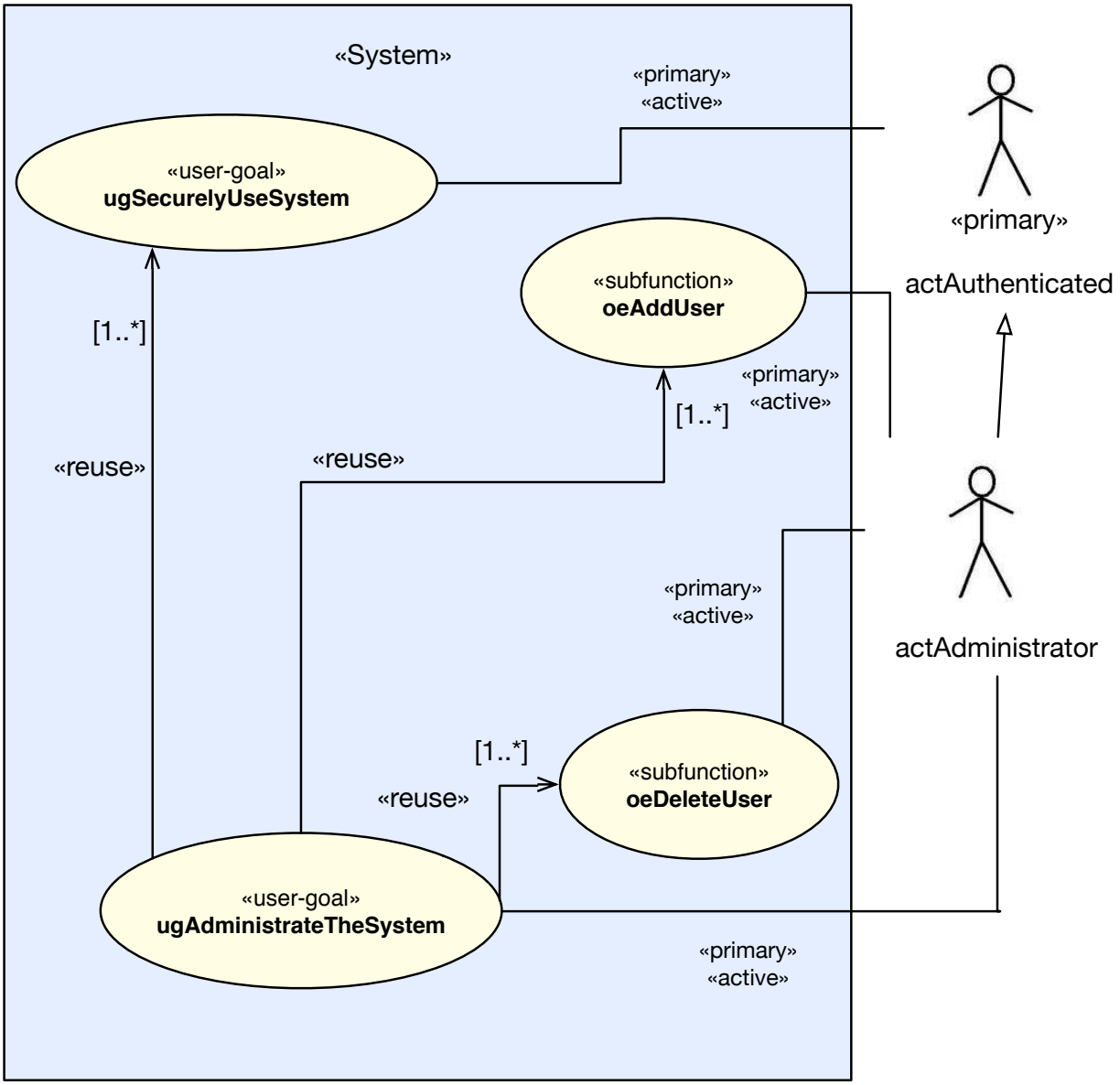


Fig. A.3 *iCrash* Use Case Diagram: ugAdministrateTheSystem

Use-Case Description	
Name	ugSecurelyUseSystem
Scope	System
Altitude	user-goal
Parameters	
1	none
Primary actor(s)	
1	`actAuthenticated[abstract]`
Secondary actor(s)	
1	none
Goal(s) description	
the goal is to follow an identification procedure and thus be allowed to safely use the system and it's features.	
Reuse	
1	`oeLogin`
2	`oeLogout`
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the a useraccount has been created for the user trying to securely use the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the `actAuthenticated` is known by the system not to be logged.
Main success steps	
a	the actor `actAuthenticated` executes the `oeLogin` use case.
b	the actor `actAuthenticated` executes the `oeLogout` use case
Step Constraints Ordering and Extensions	
1	step (a) must always precede step (b).
Additional Information	
none.	

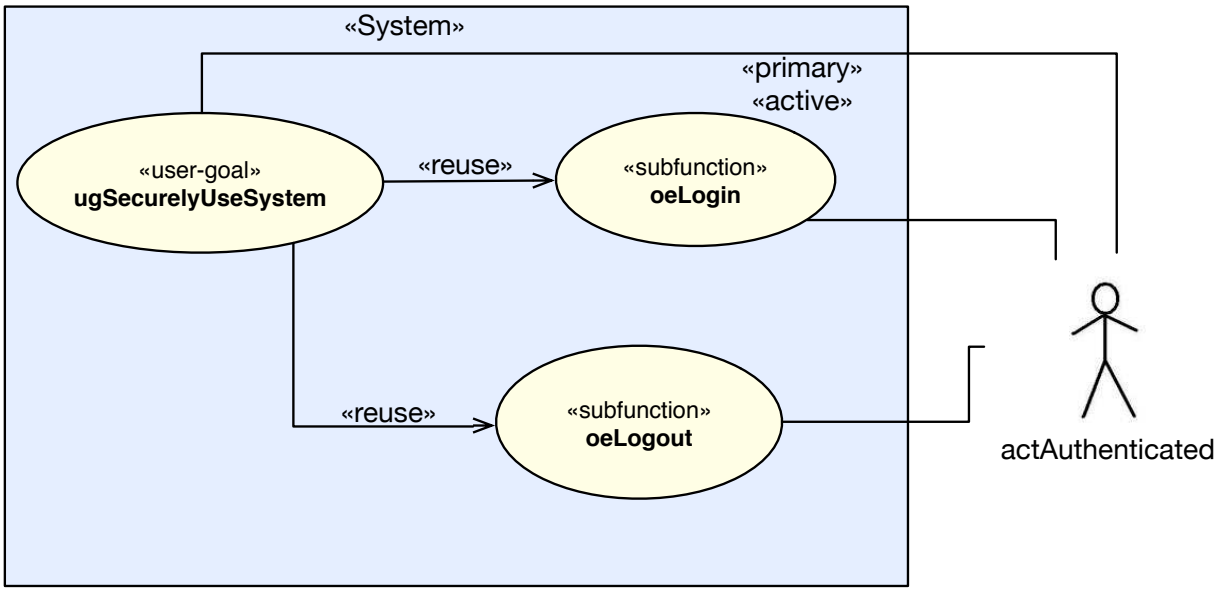


Fig. A.4 *iCrash* Use Case Diagram: ugSecurelyUseSystem

Use-Case Description	
Name	ugManageCrisis
Scope	System
Altitude	Summary
Parameters	
1	none
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	none.
Goal(s) description	
The goal is to use all operations available to the coordinator for the successful handling of a crisis or an alert received by our system.	
Reuse	
1	`oeSetCrisisStatus`
2	`oeSetCrisisHandler`
3	oeRequestEMSAssistance
4	`oeReportOnCrisis`
5	`oeCloseCrisis`
Protocol condition(s)	
1	the *iCrash* system has been started
2	the `actCoordinator` has been added to the system.
3	the `actCoordinator` has safely logged on to the system.
4	the `actDomainExpert` has validated the alert.
5	the `actCoordinator` has the right domains of expertise to manage this alert.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	there is a crisis whose related information has been changed.
Main success steps	
a	the actor `actCoordinator` executes the `oeSetCrisisHandler` use case.
b	the actor `actCoordinator` executes the `oeSetCrisisStatus` use case.
c	the actor `actCoordinator` executes the `oeReportOnCrisis` use case.
d	the actor `actCoordinator` executes the `oeCloseCrisis` use case.
Step Constraints Ordering and Extensions	
1	Step a) must be executed first.
2	Step d) must be executed last.
3	Steps b)to c) can be executed multiple times in order to manage a crisis.

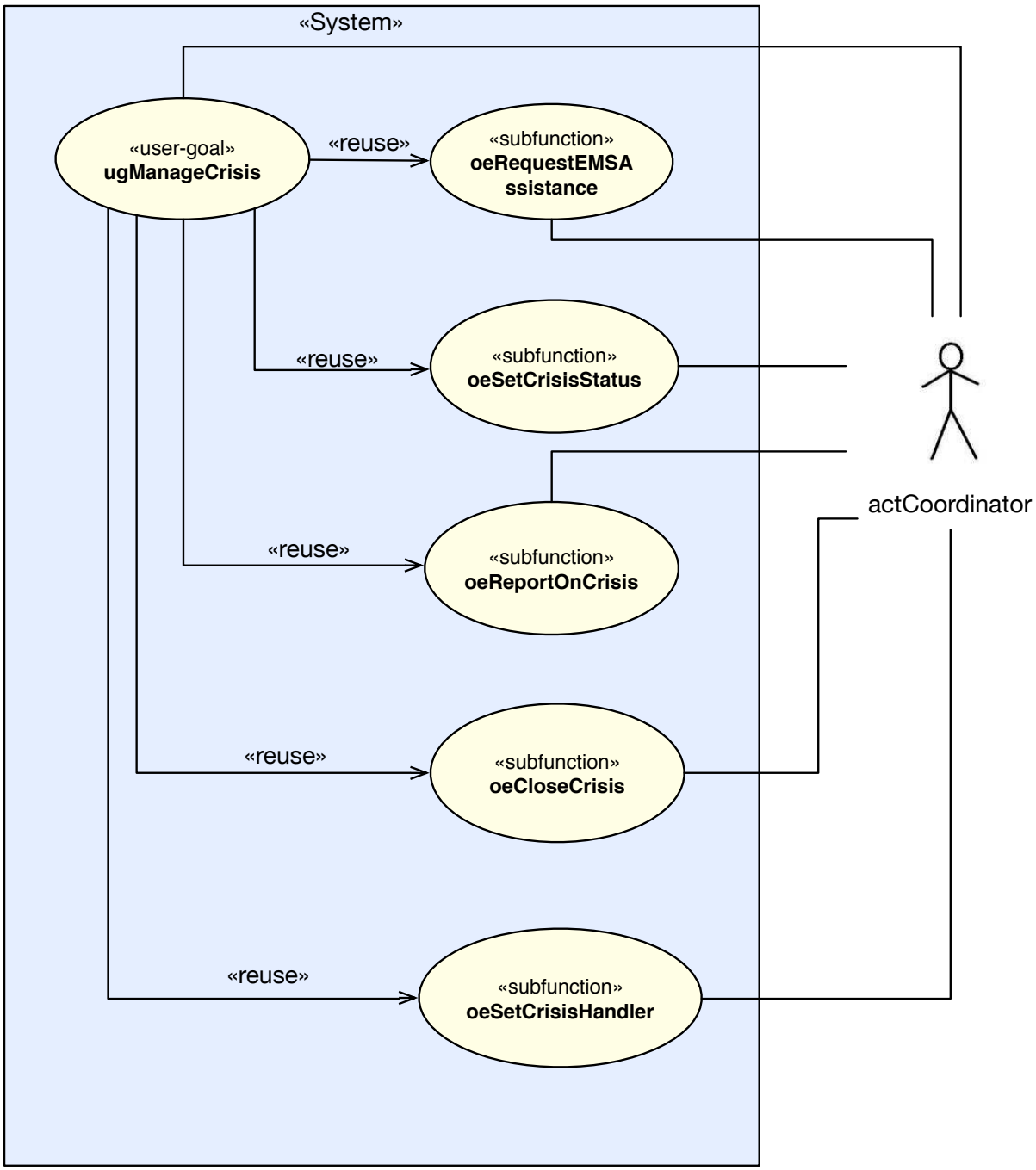


Fig. A.5 *iCrash* Use Case Diagram: UgManageCrisis

Use-Case Description	
Name	ugMonitor
Scope	System
Altitude	Summary
Parameters	
1	none
Primary actor(s)	
1	`actCoordinator[active]`
2	`actDomainExpert[active]`
Secondary actor(s)	
1	None.
Goal(s) description	
The goal is to use all operations available to the coordinator for the successful handling of a crisis or an alert received by our system.	
Reuse	
1	`oeGetAlertSet`
2	`oeValidateAlert`
3	`oeSollicitateCrisisHandling`
Protocol condition(s)	
1	the *iCrash* system has been started
2	the `actCoordinator` has been added to the system.
3	the `actCoordinator` has safely logged on to the system.
4	the `actDomainExpert` has safely logged on to system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the system is being monitored by the `actCoordinator` and `actDomainExpert`.
Main success steps	
a	the actor `actDomainExpert` executes the `oeValidateAlert` use case.
b	the actor `actCoordinator` executes the `oeGetAlertSet` use case.
c	the actor `actDomainExpert` executes the `oeSollicitateCrisisHandling` use case.
Step Constraints Ordering and Extensions	
1	Step a) must be executed first.
2	Step b) can be executed as a reaction to c) or to monitor the crisis by itself.
3	Steps a),b) and c) can be executed multiple times in order to monitor a crisis.

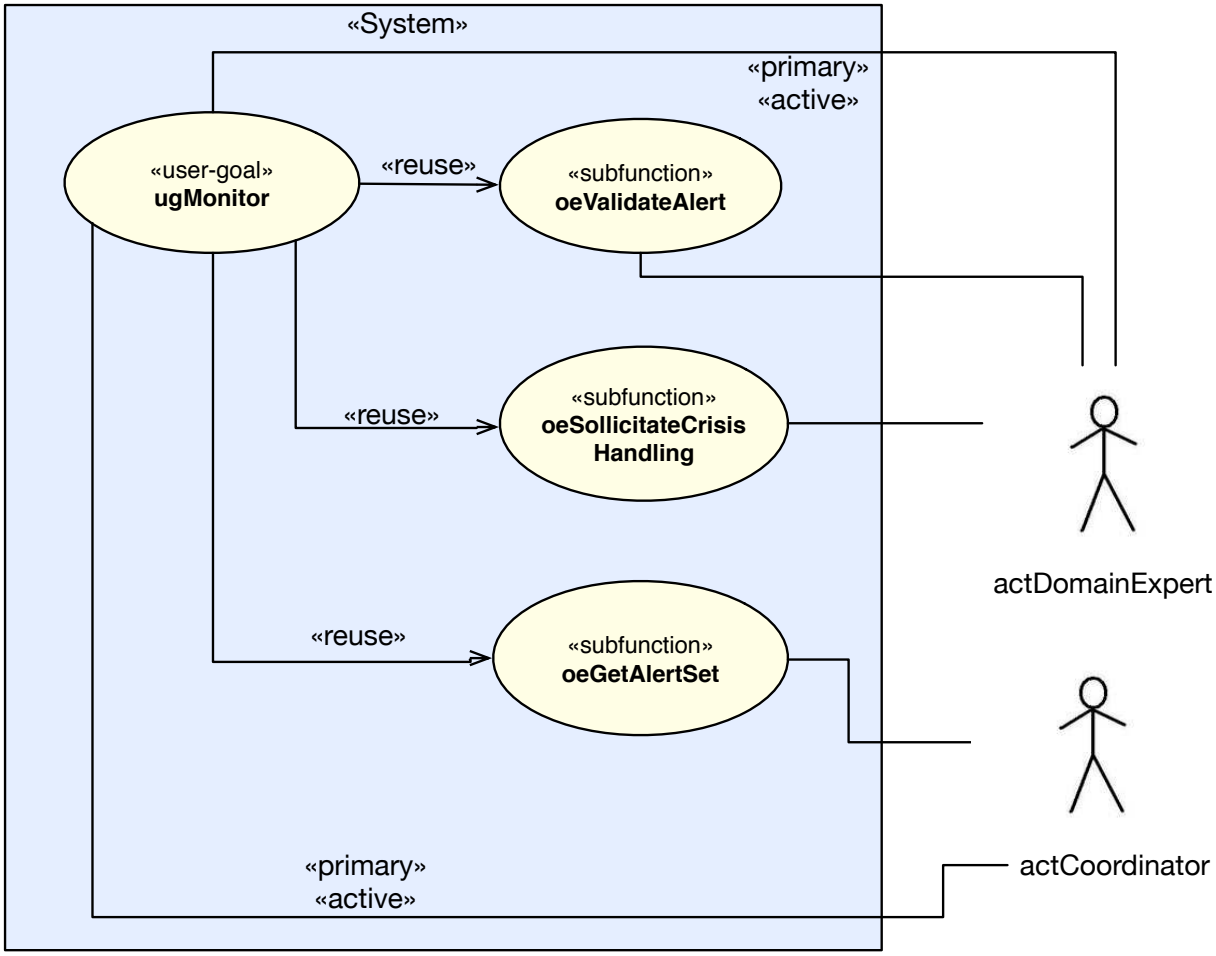


Fig. A.6 *iCrash* Use Case Diagram: UgMonitor

A.2.3 Subfunction

Use-Case Description	
Name	oeSollicitateCrisisHandling
Scope	System
Altitude	subfunction
Parameters	
1	none
Primary actor(s)	
1	`actDomainExpert[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
2	`actCoordinator[passive,multiple]`
Goal(s) description	
the `actDomainExpert`'s goal is to point out that there are crisies still not being handled by any coordinator. And he wants to change that.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	there are some crisis still pending and for which demand has been sent to the administrator and the coordinators for a certain amount of time.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a simple text message `ieMessage('Some alert have been untreated for more than the 10 min. Please REACT!')` is sent to the system administrator and to all the coordinators of the environment for each alert that is unhandled and for which no demand has been sent to the administrator and the coordinators for more than a certain amount of time.
2	the reminder period for the concerned crisis is initialized.
Main success steps	
a	the actor `actDomainExpert` sends the message `oeSollicitateCrisisHandling()` to the system.
Step Constraints Ordering and Extensions	
1	none.
Additional Information	
none	

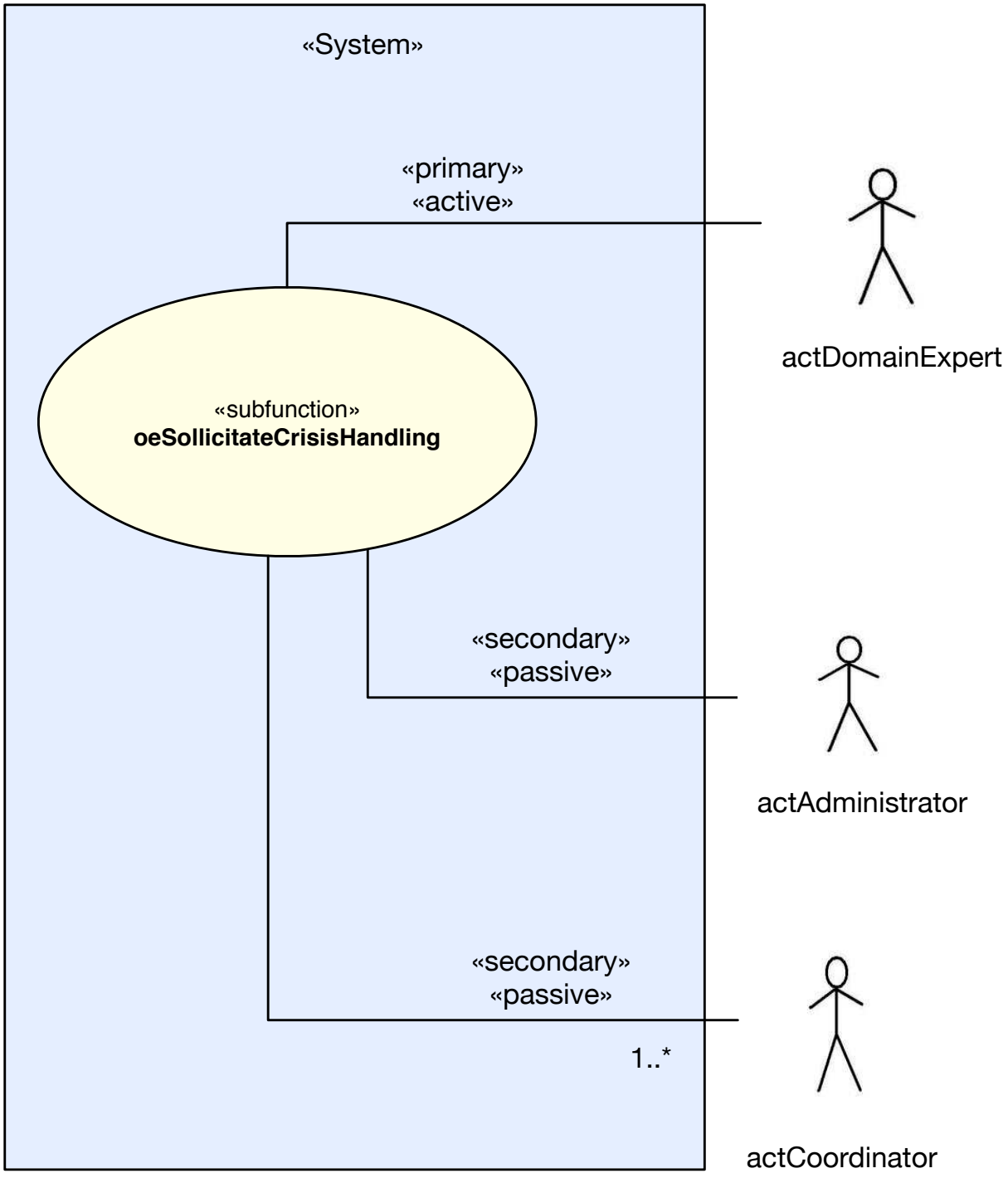


Fig. A.7 *iCrash* Use Case Diagram: oeSollicitateCrisisHandling

Use-Case Description	
Name	oeAddUser
Scope	System
Altitude	subfunction
Parameters	
1	`UserID:dtUserID` - the user's identification.
2	`Login:dtLogin` - the username of the user.
3	`Password:dtPassword` - the password of the user.
4	`TokenID:dtTokenID` - the a unique identifier for the users token.
5	`Usertyep:etUserType` - the Type of user [Coordinator, DomainExpert or EMS]
6	`CoordinatorDomain:ctDomain` - the domains of expertise of a coordinator.
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
2	`actDomainExpert[passive]`
3	`actEMS[passive]`
Goal(s) description	
the `actAdministrator`'s goal is to delete new user to the system so he/she can safely use the system.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A new user has been added to the system.
2	The new user can use the system safely.
Main success steps	
a	the actor `actAdmnistrator` sends the message `oeAddUser(CoordinatorID,Login,Password,TokenI` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
When creating Coordinator you must specify his domains of expertise. Only users of the type Coordinator need to have specific Domains.	

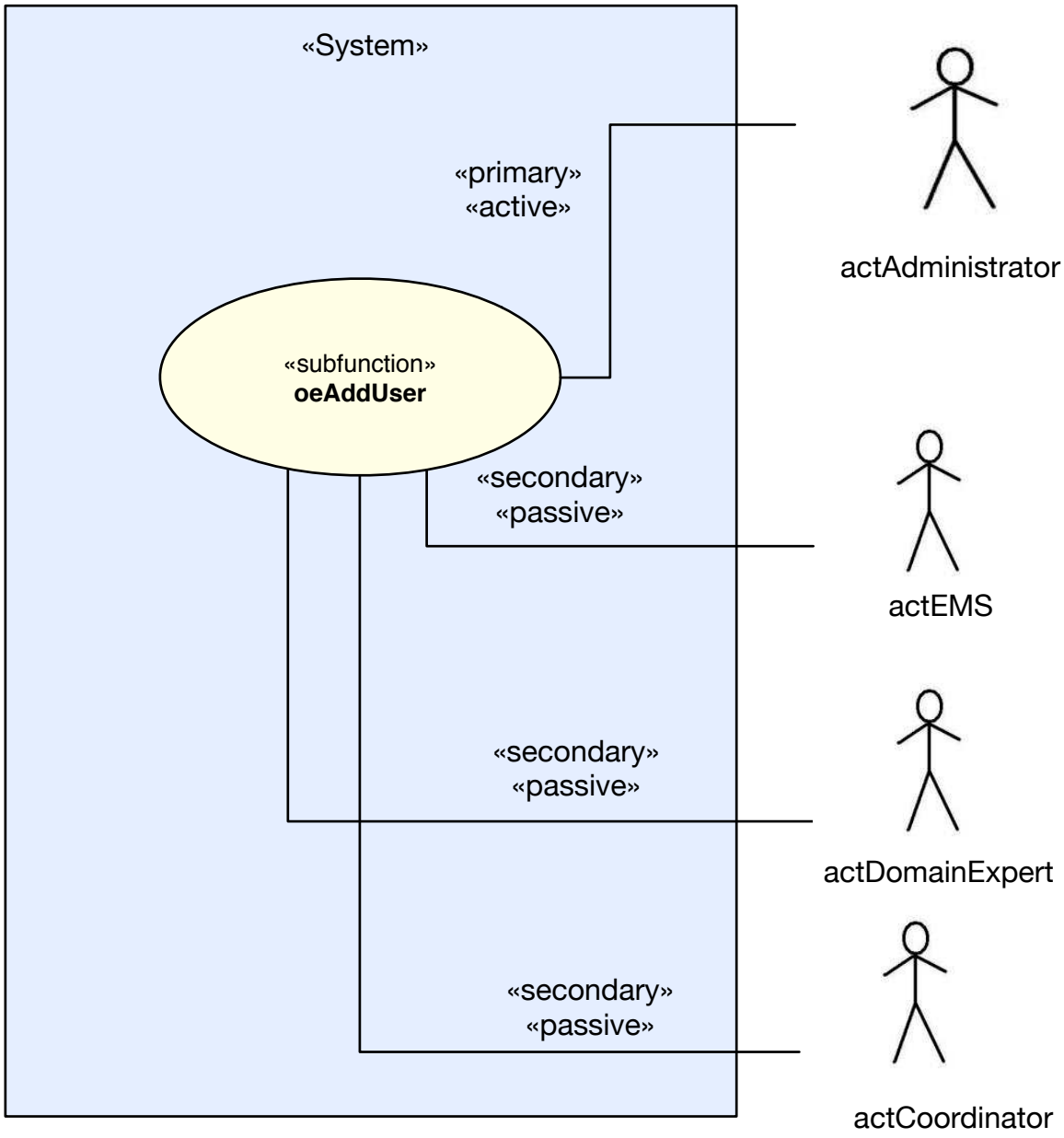


Fig. A.8 *iCrash* Use Case Diagram: oeAddUser

Use-Case Description	
Name	oeDeleteUser
Scope	System
Altitude	subfunction
Parameters	
1	`UserID:dtUserID` - the user's identification
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
2	`actDomainExpert[passive]`
3	`actEMS[passive]`
Goal(s) description	
the `actAdministrator`'s goal is to remove a user from the system.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the user with the specified user ID already exists in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A user has been removed from the system.
2	The specified user can no longer log on to the system.
Main success steps	
a	the actor `actAdmnistrator` sends the message `oeDeleteUser(UserID)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	

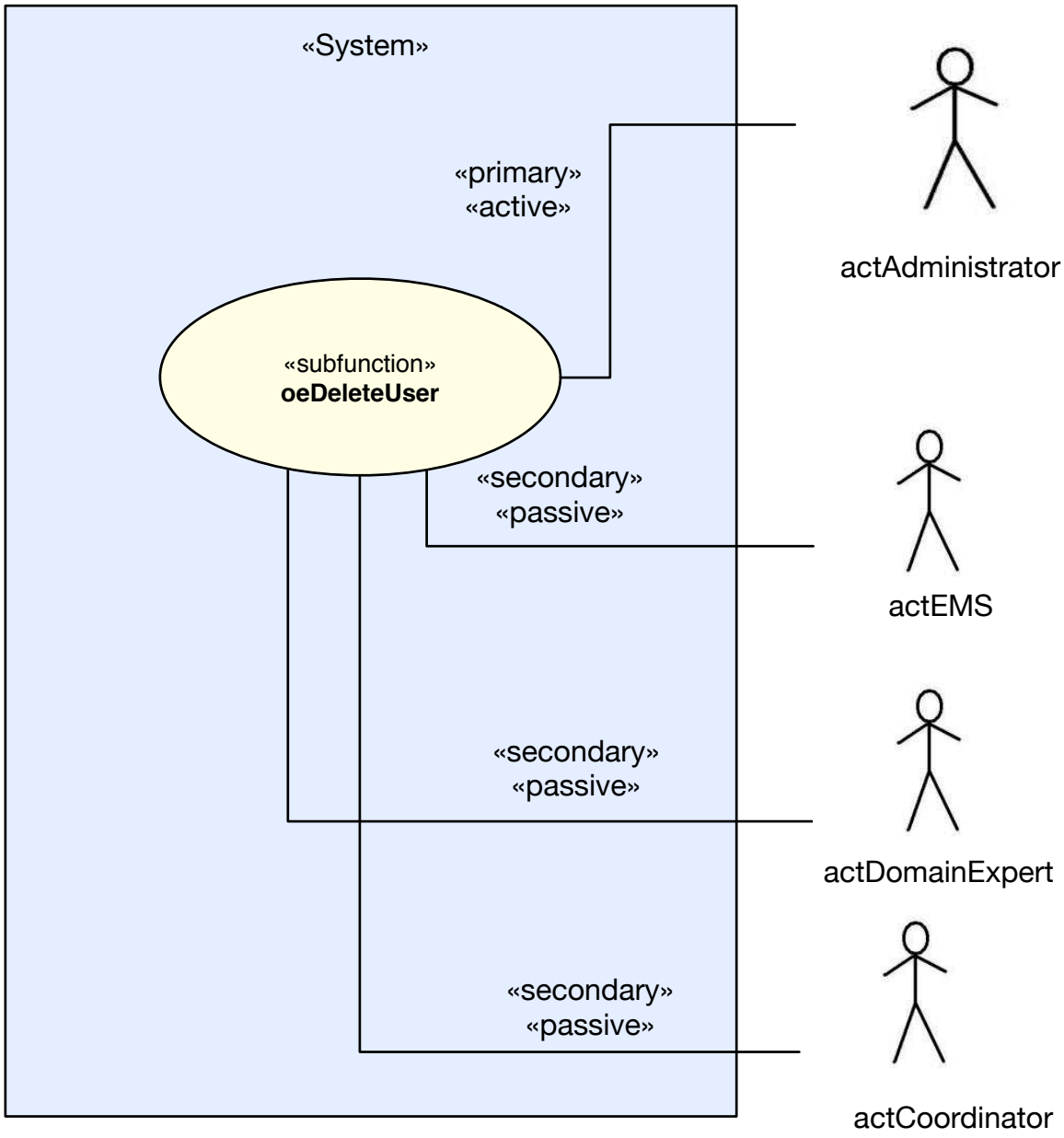


Fig. A.9 *iCrash* Use Case Diagram: oeDeleteUser

Use-Case Description	
Name	oeRequestEMSAssistance
Scope	System
Altitude	subfunction
Parameters	
1	`Message:dtComment` - A message from the `actCoordinator` to the `actEMS`
2	`UniqueIDOftheReq:dtRequestID` - a request identification enabling`actCoordinators` and `actEMSs` to differentiate requests.
3	`NumberofVehiculesinaccident:dtNbrofVehiculesInAccident` - the number of vehicles involved in the accident.
4	`NumberofVictimsinaccident:dtNbrofVictims` - the number of victims involved in the accident.
5	`TypeOfreqEMSAssistance:ctEMSType` - the type of EMS unit requested.(RequestFireFighter, RequestPolice and RequestAmbulance)You can ether request one of each or just two or all.
Primary actor(s)	
1	`Coordinator[active]`
Secondary actor(s)	
1	None.
Goal(s) description	
the `actCoordinator`'s goal is to request a EMS unit to provide assistance to the victims of a car accident.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	an Alert has been received by our system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A message is sent to the `actEMS` requesting assistance.
Main success steps	
a	the actor `actCoodinator` sends the message`oeRequestEMSAssistance(dtComment AdtComment, dtRequestID AdtRequest, dtGPSLocation AdtGPSLocation, dtNbrofVehiculesInAccident AdtVehiculesInAccident, dtNbrOfVictims AdtNbrVictims, ctEMStype ActEMSType)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
The coordinator decided ultimately what information is put into the message to the EMS actor the shown data types are only meant as an example of what can be added to such a message.	

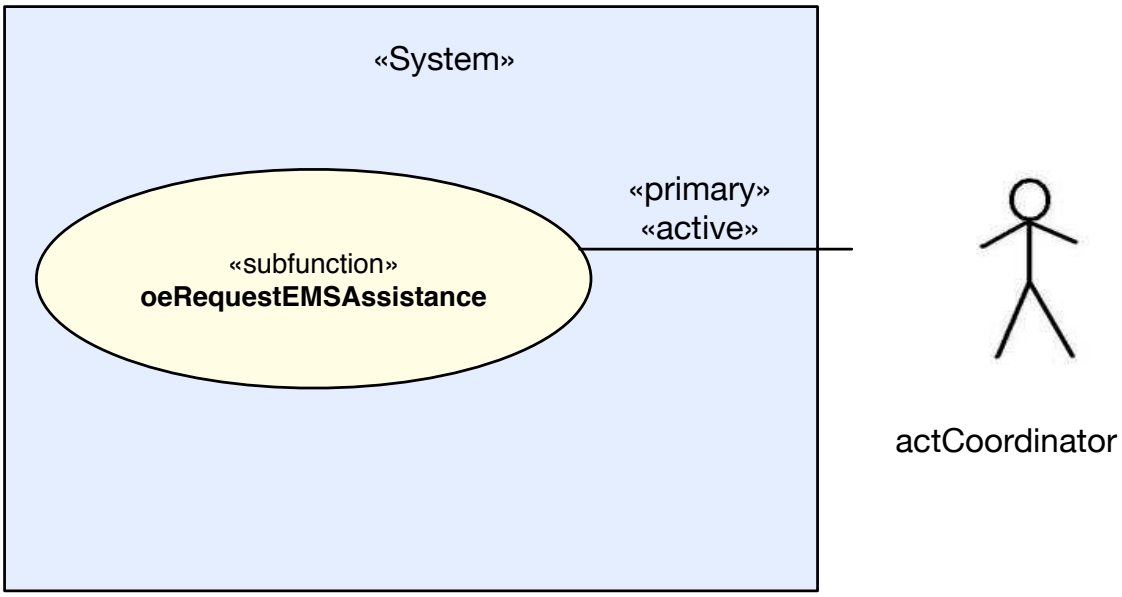


Fig. A.10 *iCrash* Use Case Diagram: oeRequestEMSAssistance

Use-Case Description	
Name	oeSollicitateCrisisHandling
Scope	System
Altitude	subfunction
Parameters	
1	none
Primary actor(s)	
1	`actDomainExpert[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
2	`actCoordinator[passive,multiple]`
Goal(s) description	
the `actDomainExpert`'s goal is to point out that there are crisies still not being handled by any coordinator. And he wants to change that.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	there are some crisis still pending and for which demand has been sent to the administrator and the coordinators for a certain amount of time.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a simple text message `ieMessage('Some alert have been untreated for more than the 10 min. Please REACT!')` is sent to the system administrator and to all the coordinators of the environment for each alert that is unhandled and for which no demand has been sent to the administrator and the coordinators for more than a certain amount of time.
2	the reminder period for the concerned crisis is initialized.
Main success steps	
a	the actor `actDomainExpert` sends the message `oeSollicitateCrisisHandling()` to the system.
Step Constraints Ordering and Extensions	
1	none.
Additional Information	
none	

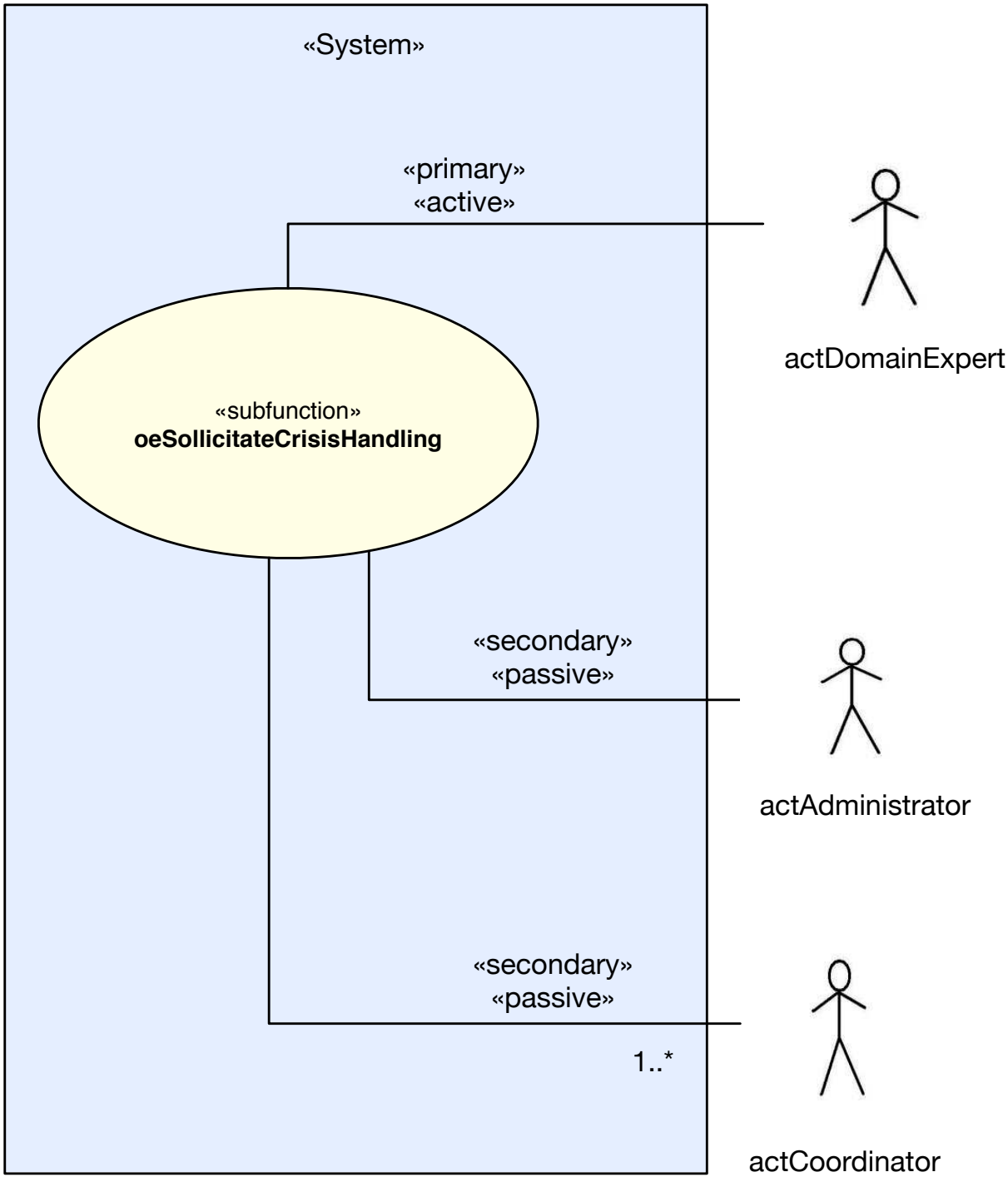


Fig. A.11 *iCrash* Use Case Diagram: oeSollicitateCrisisHandling

Use-Case Description	
Name	oeReportOnCrisis
Scope	System
Altitude	subfunction
Parameters	
1	`AlertID:dtAlertID` - the identification of the Alert.
2	`AText:dtComment` - a text written by the actCoordinator to better describe the crisis, in his own words, in the report.
3	`UserID:dtUserID` - a unique identifier identifying the `actCoordinator` writing the report.
4	`TypeofCrisis:etCrisisType` - the type of the crisis differentiated here by (Small, Medium or Huge) thus describing the crisis type.
5	`CrisisStatus:etCrisisStatus` - the status of the crisis differentiated here by (pending, handled or solved).
6	`RecievedMessages:dtRecievedMessage` - This is a record of the alert messages received from the actor `actComCompany`.
7	`NumberofVehiculesinaccident:dtNbrofVehiculesInAccident` - the number of vehicles involved in the accident.
8	`NumberofVictimsinaccident:dtNbrofVictims` - the number of victims involved in the accident.
Primary actor(s)	
1	`Coordinator[active]`
Secondary actor(s)	
1	`ComCompany[passive]`
Goal(s) description	
the `actCoordinator`'s goal is to write a report about the current ongoing crisis or to write a final report of the crisis.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	an Alert has been received by our system.
3	the `actDomainExpert` has validated the alert.
4	the `actCoordinator` actor has the right domain to access the Crisis and is thus allowed to write a report about it.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A report of a certain crisis exists in the system.
Main success steps	
a	the actor `actCoodinator` sends the message`oeReportOnCrisis(dtAlertID AdtCrisisID, dtComment AdtComment, dtUserID AdtUserID, etCrisisType AdtCrisisType, etCrisisStatus AdtCrisisStatus, dtRecievedMessage AdtReceivedMessage, dtNbrofVehiculesInAccident AdtVehiculesInAccident, dtNbrOfVictims AdtNbrVictims)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
The coordinator decided ultimately what information is put into the report the shown data types are only meant as an example of what can be added to such a report.	

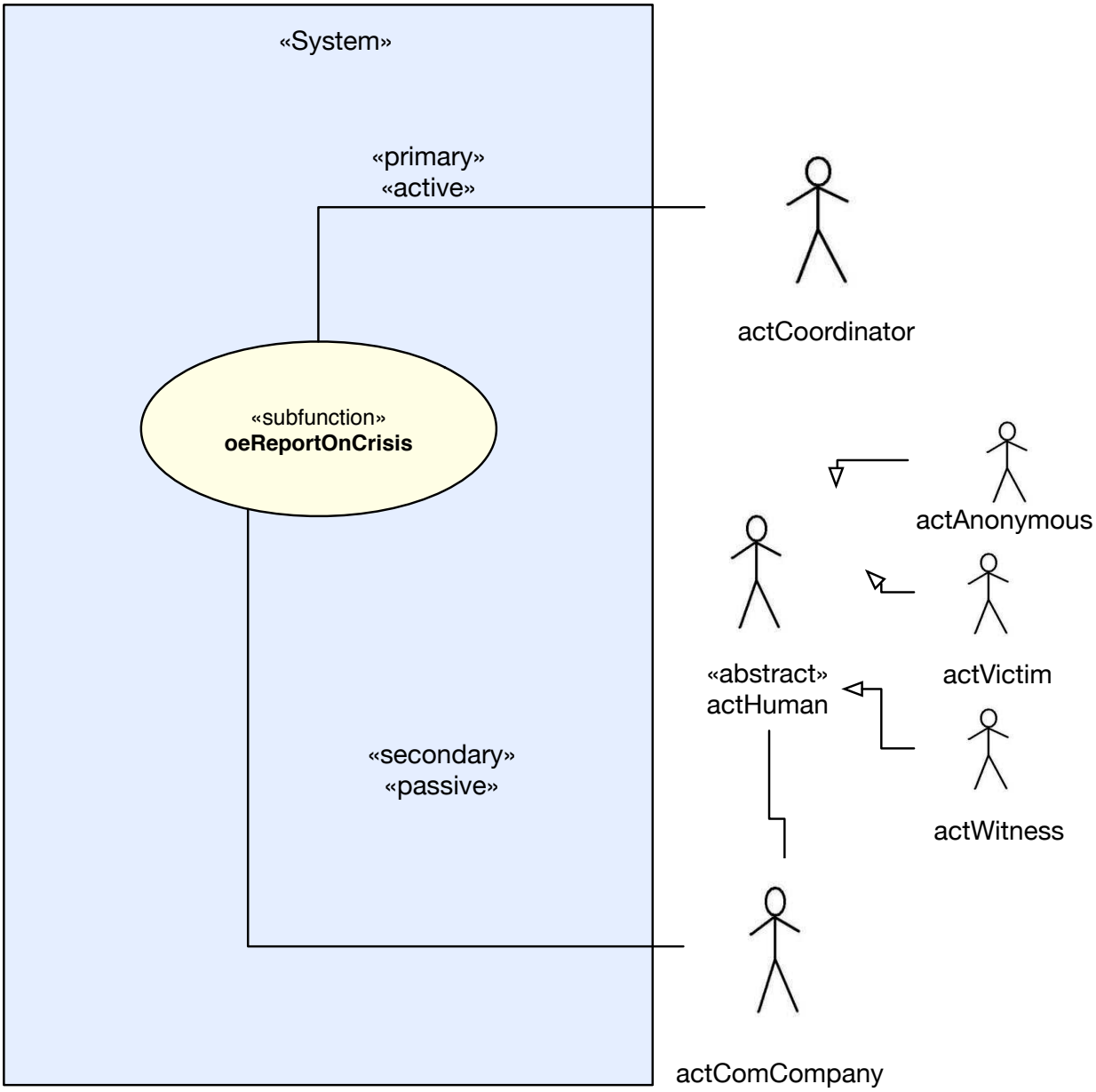


Fig. A.12 *iCrash* Use Case Diagram: oeReportOnCrisis

Use-Case Description	
Name	oeInfoFam
Scope	System
Altitude	subfunction
Parameters	
1	`Message:dtComment` - the message that will be sent to the family members.
2	`VictimDescription:ctVictim` - included in this data are all the information about the victim. This includes the victims Name, phone number as well as the ICE contacts phone number and name.
3	`NameofHospital:dtHospitalName` - the name of the hospital that the victim was brought to.
Primary actor(s)	
1	`ComCompany[active]`
Secondary actor(s)	
1	`actCoordinator`
Goal(s) description	
the `actComcompany`'s goal is send a message edited by the `actCoordinator`.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the `actEMS` reports that a victim was delivered to the hospital.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A message is sent to a victims ICE contact.
Main success steps	
a	the actor `actCoodinator` sends the message `oeInfoFam(dtComment AdtComment, ctVictim ActVictim dtHospitalName AdtHospitalName)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
The coordinator ultimately decides what information is put into the message to the family members the shown data types are only meant as an example of what can be added to such a message.	

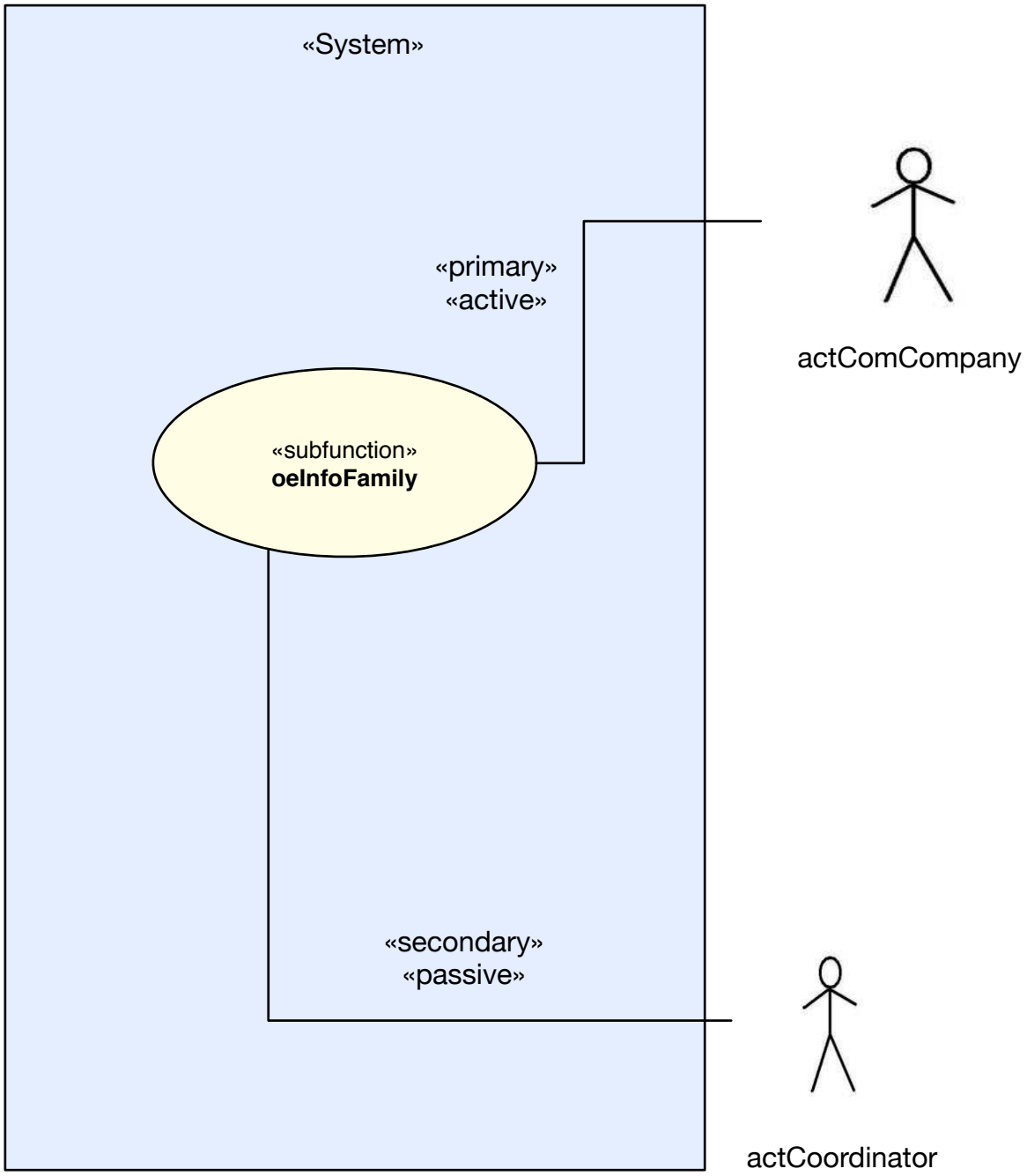


Fig. A.13 *iCrash* Use Case Diagram: oeInfoFamily

Use-Case Description	
Name	oeCloseCrisis
Scope	System
Altitude	subfunction
Parameters	
1	`crisisID:dtCrisisID` - the crisis identification
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actComCompany[passive]`
Goal(s) description	
the `actCoordinator`'s goal is to declare a crisis as closed.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	a crisis with the given ID exsits in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the crisis is closed.
2	a message `ieSmsSend(AdtPhoneNumber,AdtComment)` is sent to the `actComCompany` actor for each registered alert declared by a SMS coming from phone number `AdtPhoneNumber` and related to the crisis `crisisID`. All the messages have the same textual message `AdtComment` giving them the minimal information on the crisis they alerted.
Main success steps	
a	the actor `actCoordinator` sends the message `oeCloseCrisis(crisisID)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
none	

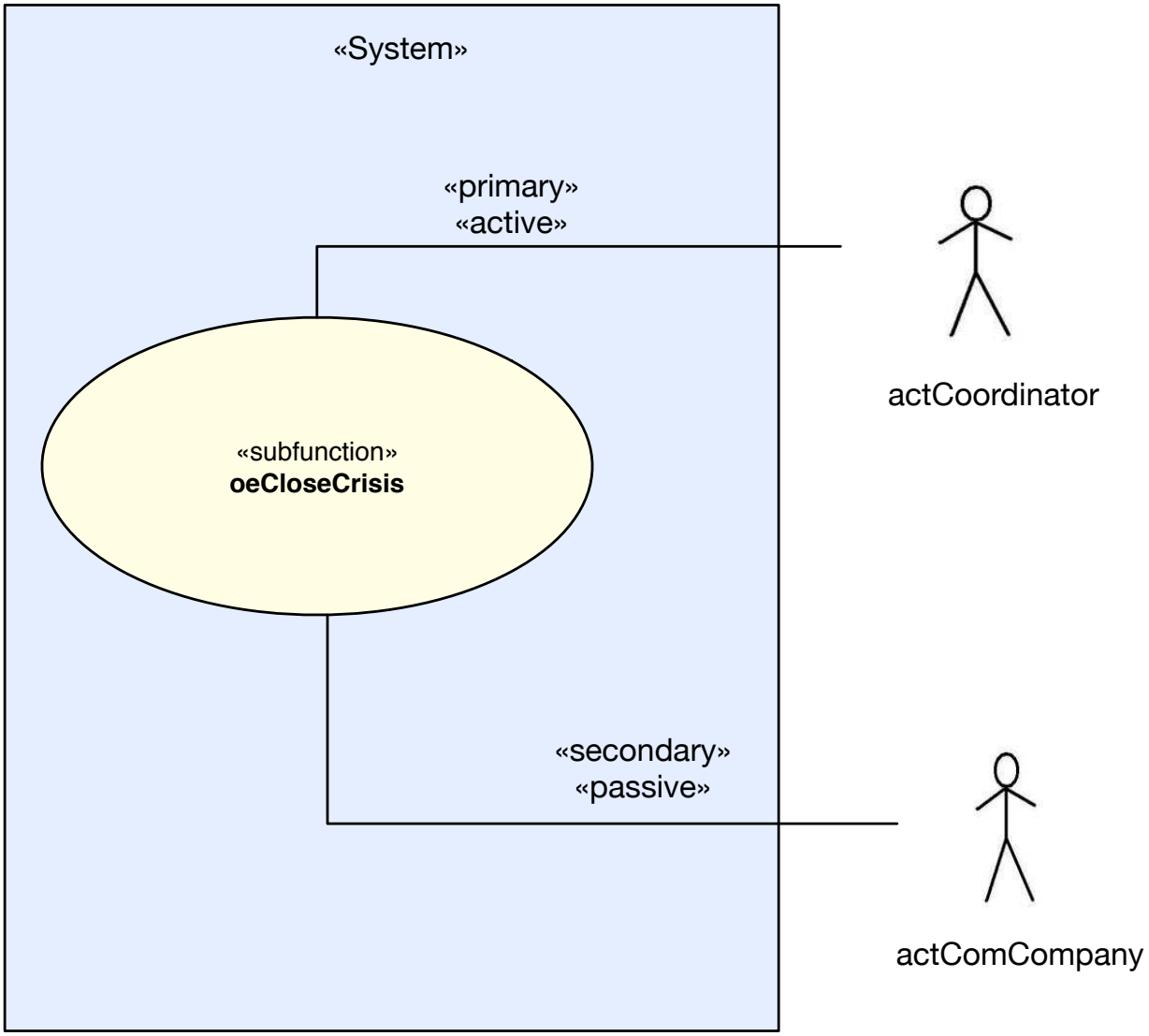


Fig. A.14 *iCrash* Use Case Diagram: oeCloseCrisis

Use-Case Description	
Name	oeAlert
Scope	System
Altitude	subfunction
Parameters	
1	`HumanKind:etHumanKind` - the type of individual trying to use the system(victim, witness or anonymous).
2	`Date:dtDate` - the date the Alert was sent.
3	`Time:dtTime` - the time the Alert was sent.
4	`PhoneNumber:dtPhoneNumber` - the phone number from which the Alert was sent.
5	`GPSLocation:dtGPSLocation` - the GPS location the Alert originated from.
6	`Comment:dtComment` - a text meant for the human to describe his situation.
Primary actor(s)	
1	`actComCompany[active]`
Secondary actor(s)	
1	`actDomainExpert[passive]`
Goal(s) description	
the `actComCompany`'s goal is to send an Alert from a human to the system.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	An Alert is received by the system.
Main success steps	
a	the actor `actComCompany` sends the message `oeAlert(etHumanKind, dtDate, dtTime, dtPhoneNum` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	

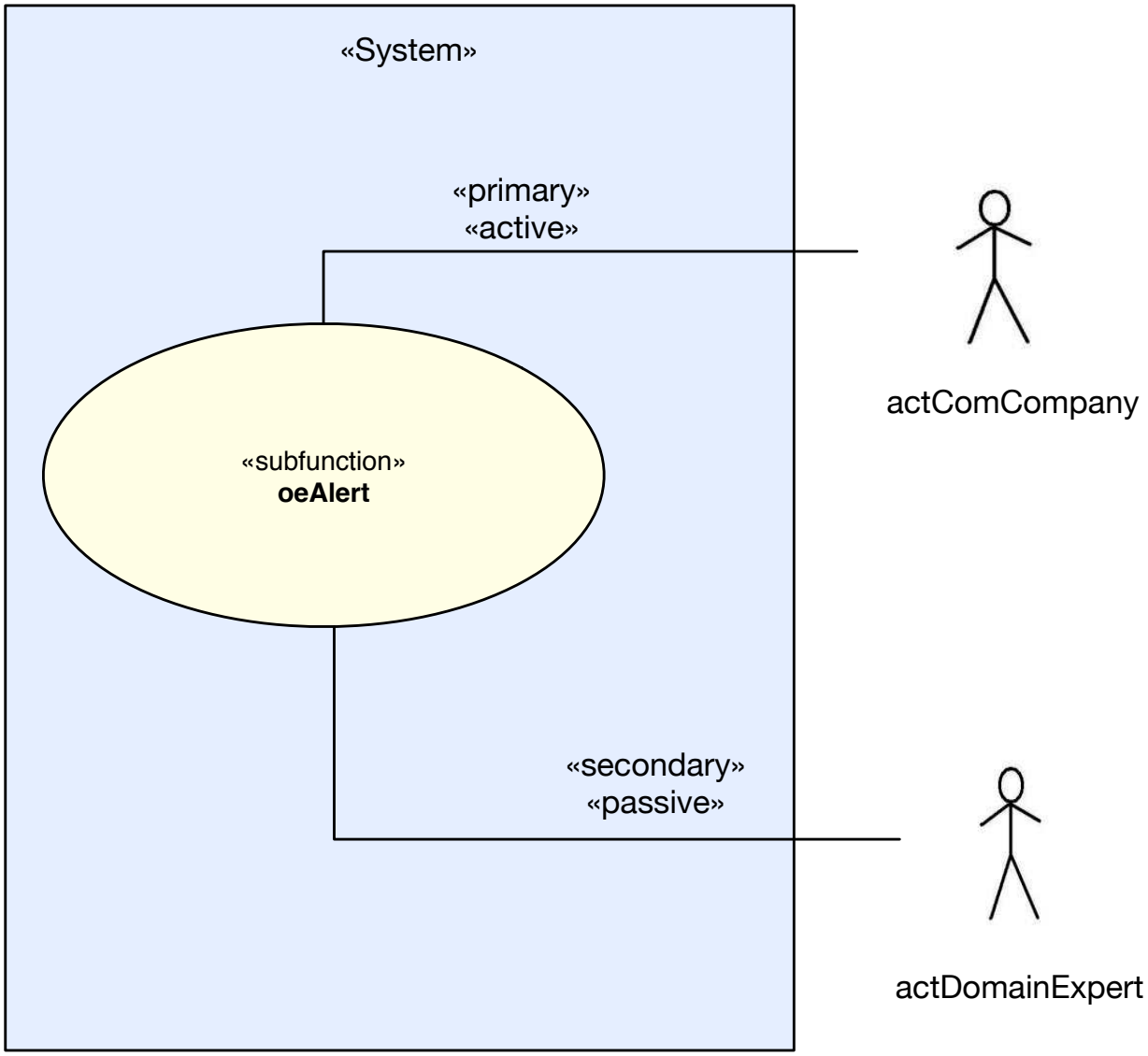


Fig. A.15 *iCrash* Use Case Diagram: oeAlert

Use-Case Description	
Name	oeGetAlertSet
Scope	System
Altitude	subfunction
Parameters	
1	`AlertStatus:etAlertStatus` - the Alert status.
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actDomainExpert[passive]`
Goal(s) description	
the `actCoordinator`'s goal is to get a list/set of all the Alerts of a certain alert-status.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the `actCoordinator` has logged on to the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the `actCoordinator` is now aware of all the Alerts of a specified status.
Main success steps	
a	the actor `actCoordinator` sends the message `oeCloseCrisis(crisisID)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
none	

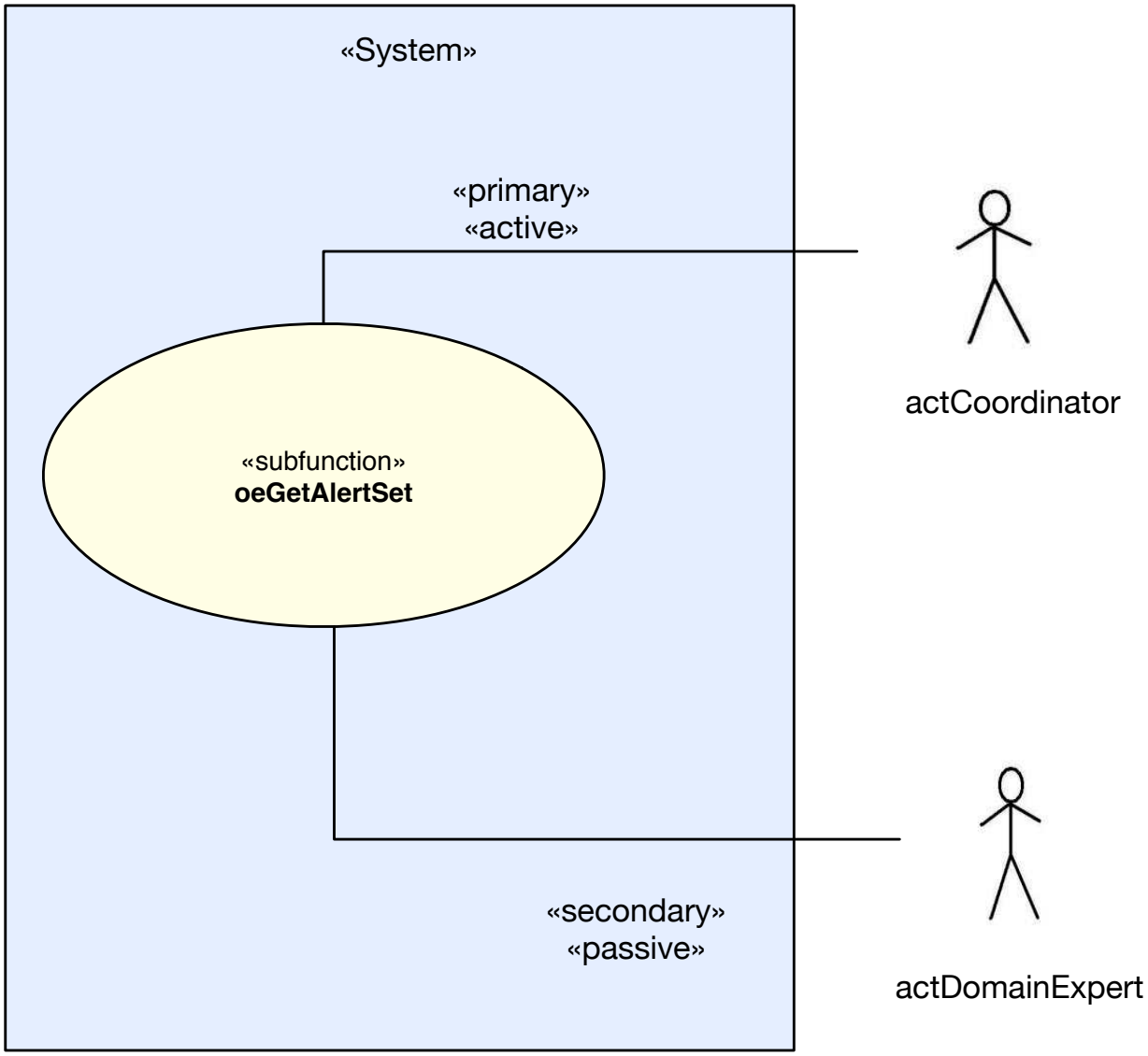


Fig. A.16 *iCrash* Use Case Diagram: oeGetAlertSet

Use-Case Description	
Name	oeSetCrisisHandler
Scope	System
Altitude	subfunction
Parameters	
1	`AlertID:dtAlertID` - the Alert's unique identification.
2	`Crisisdata:ctCrisis` - the data required to initiate a crisis(CrisisType, CrisisStatus, GPSLocation, Date and Time).
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	None.
Goal(s) description	
the `actCoordinator`'s goal is to indicate to the system that he will take care of a certain Alert and thus create a crisis in the system.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the `actCoordinator` has logged on to the system.
3	the `actCoordinator` has to have the right domains of expertise to be allowed access to the crisis.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	A crisis has been created for a certain Alert.
2	the `actCoordinator` has indicated to the system that he will now take care of the Alert sepecified.
Main success steps	
a	the actor `actCoordinator` sends the message `oeSetCrisisHandler(AlertID,ctCrisis)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
none	

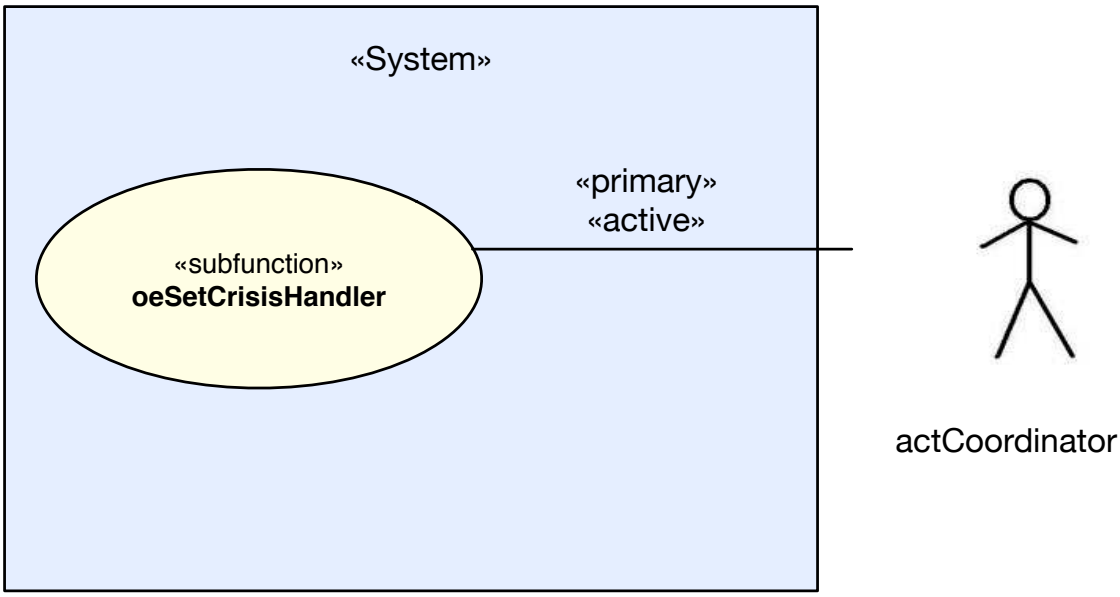


Fig. A.17 *iCrash* Use Case Diagram: oeSetCrisisHandler

Use-Case Description	
Name	oeLogin
Scope	System
Altitude	subfunction
Parameters	
1	`Username:dtLogin` - the user's login/username.
2	`Password:dtPassword` - the user's password.
3	`SerialKey:dtKey` - the user's serial key generated by his token device.
Primary actor(s)	
1	`actAuthenicated[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
2	`actCoordinator[passive]`
3	`actDomainExpert[passive]`
4	`actEMS[passive]`
Goal(s) description	
the `actAuthenticated`'s to log on to the system in order to securely use it.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the `actAuthenicated` has securely logged on to the system.
Main success steps	
a	the actor `actAuthenticatedActor` sends the message `oeLogin(dtLogin, dtPassword, dtKey)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	

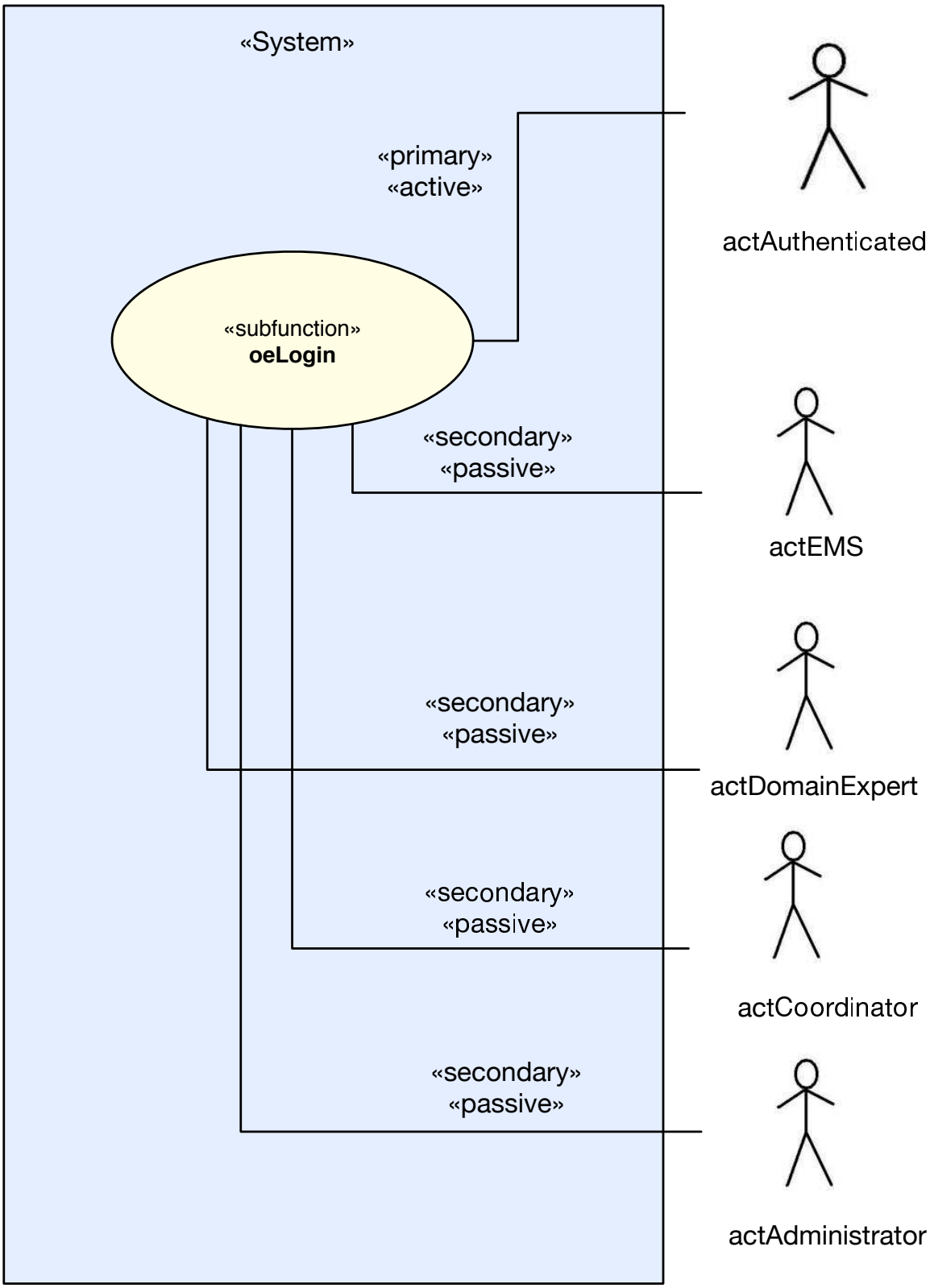


Fig. A.18 *iCrash* Use Case Diagram: oeLogin

Use-Case Description	
Name	oeLogout
Scope	System
Altitude	subfunction
Parameters	
1	None.
Primary actor(s)	
1	`actAuthenicated[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
2	`actCoordinator[passive]`
3	`actDomainExpert[passive]`
4	`actEMS[passive]`
Goal(s) description	
the `actAuthenticated`'s to logoff from the system.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	the `actAuthenticated` performing the opperation is logged in.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the `actAuthenicated` has securely logged off from the system.
Main success steps	
a	the actor `actAuthenticatedActor` sends the message `oeLogoff()` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	

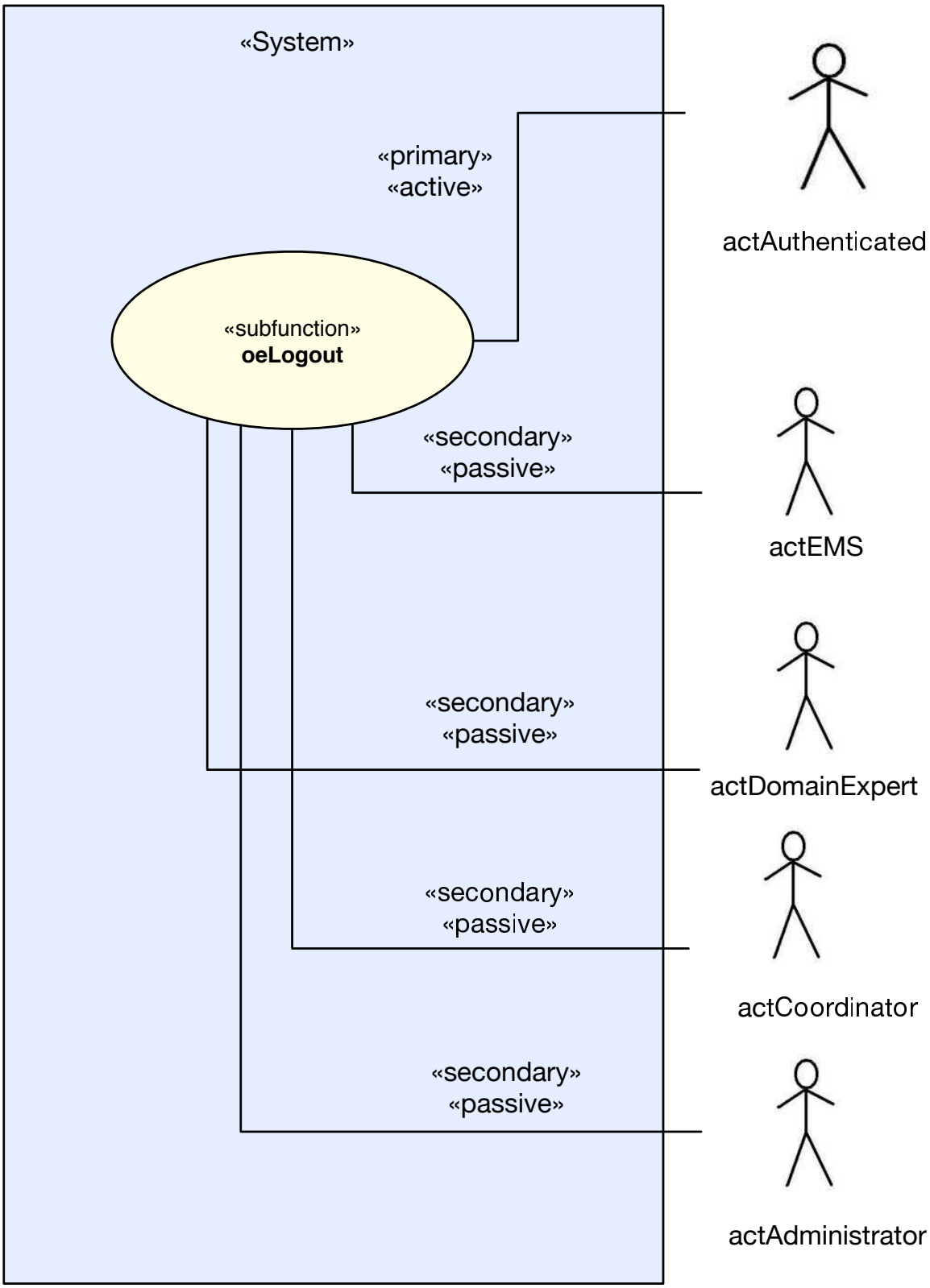


Fig. A.19 *iCrash* Use Case Diagram: oeLogout

Use-Case Description	
Name	oeReportEMSCrisisStatus
Scope	System
Altitude	subfunction
Parameters	
1	`crisisID:dtCrisisID` - the crisis identification.
2	`Comment:dtComment` - a comment field for the EMS actor to describe the situation.
3	`EMSCrisisStatus:etEMSCrisisStatus` - the status of the situation(here crisis) as seen from their perspective.
4	`HospitalName:dtHospitalName` - the name of the hospital.
5	`Victim information:ctVictim` - information about the victim.(VictimName,VictimPhonnumber,HumanKind, VictimICEContactName and VictimICEPhoneNumber)
Primary actor(s)	
1	`actEMS[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
Goal(s) description	
the `actEMS`'s goal is to describe the status of the situation they were asked to provide assistance to.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	a crisis with the given ID exsits in the system.
3	an `actCoordinator` has requested EMS assistance for the specified crisis.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the report of the current status of the crisis from the `actEMS`'s point of view has been sent to the system.
Main success steps	
a	the actor `actEMS` sends the message `oeReportEMSCrisisStatus(dtCrisisID, dtComment, dtEMSCr` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
This is just an example of data types that might be used in a message to report the ems crisis status, ultimately the `actEMS` decides what he puts in to such a message.	

Fig. A.20 *iCrash* Use Case Diagram: oeCloseCrisis

Use-Case Description	
Name	oeReplyToRequest
Scope	System
Altitude	subfunction
Parameters	
1	`crisisID:dtCrisisID` - the crisis identification.
2	`Comment:dtComment` - a comment field for the EMS actor to describe the situation.
3	`RequestID:dtRequestID` - a unique identifier for a request.
4	`EMSCrisisStatus:etEMSCrisisStatus` - the status of the situation(here crisis) as seen from their perspective.
Primary actor(s)	
1	`actEMS[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
Goal(s) description	
the `actEMS`'s goal is to reply to a request previously sent by a coordinator.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	a crisis with the given ID exsits in the system.
3	an `actCoordinator` has requested EMS assistance for the specified crisis.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the `actEMS` has sent a reply to a received request to the system.
Main success steps	
a	the actor `actEMS` sends the message `oeReplyToRequest(dtCrisisID, dtComment, dtEMSCrisisSta` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
This is just an example of data types that might be used in a message to reply to a request ultimately the `actEMS` decides what he puts in to such a message.	

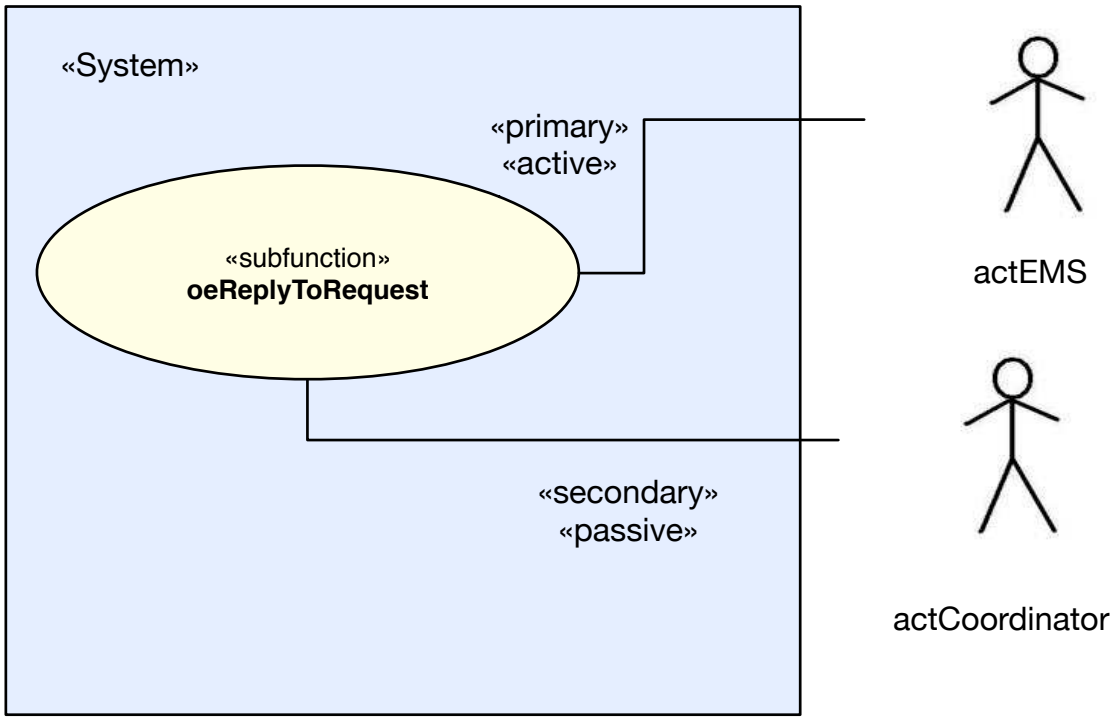


Fig. A.21 *iCrash* Use Case Diagram: oeReplyToRequest

Use-Case Description	
Name	oeSetCrisisStatus
Scope	System
Altitude	subfunction
Parameters	
1	`crisisID:dtCrisisID` - the crisis identification.
2	`CrisisStatus:etCrisisStatus` - the status of the crisis.(in-handling, handled and solved)
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actComCompany[passive]`
Goal(s) description	
the `actCoordinator`'s goal is to change the status of a crisis.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	a crisis with the given ID exists in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the status of the crisis has changed.
Main success steps	
a	the actor `actCoordinator` sends the message `oeSetCrisisStatus(dtCrisisID, etCrisisStatus)` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
none	

Fig. A.22 *iCrash* Use Case Diagram: oeCloseCrisis

Use-Case Description	
Name	oeValidateAlert
Scope	System
Altitude	subfunction
Parameters	
1	`AlertID:dtAlertID` - the Alert unique identification.
2	`GPSLocation:dtGPSLocation` - the GPS location of the Alert.
3	`DateAndTime:dtDateAndTime` - the date and time of when the alert was sent.
4	`AlertStatus:etAlertStatus` - the Alert status.
5	`AlertDomains:dtDomain` - Domains of expertise required to handle this Alert.
Primary actor(s)	
1	`actDomainExpert[active]`
Secondary actor(s)	
1	None.
Goal(s) description	
the `actDomainExpert`'s goal is to verfy that an Alert is valid and to assign Domains to that Alert in order select the right Coordinator handle it.	
Reuse	
1	none
Protocol condition(s)	
1	the *iCrash* system has been deployed.
2	an Alert with the given ID exists in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	an Alert's status has been changed to verifed and given Domains of expertise required to handle it.
Main success steps	
a	the actor `actDomainExpert` sends the message `oeVerifyAlert(dtAlertID, dtGPSLocation, dtDat` to the system.
Step Constraints Ordering and Extensions	
1	none
Additional Information	
none	

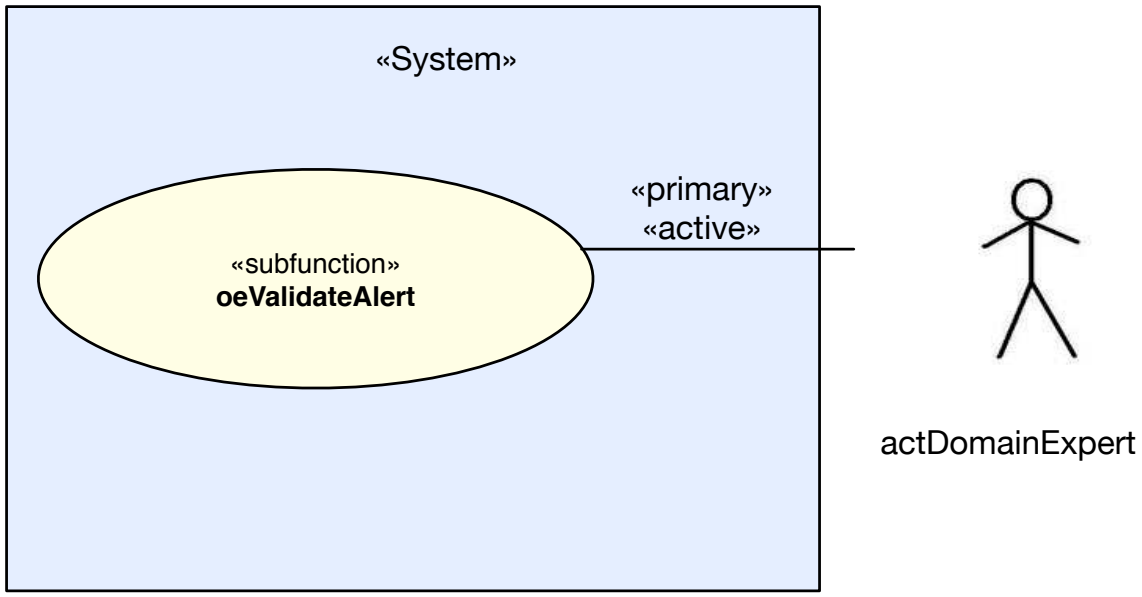


Fig. A.23 *iCrash* Use Case Diagram: oeValidateAlert

A.3 Use case instance Table(s)

USE-CASE INSTANCE

Name
Simple and Complete DeployAndRun Instance

Instance ID
01

Environmental conditions and assumptions
The necessary IT infrastructure exists to allow for deployment of the *iCrash* system.

Inputs
inputs are sequences of characters interpreted as string or numbers.

Instance flow description

1 the `actMsrCreator` actor `theCreator` sends the message `oeCreateSystemAndEnvironment(1)` requesting the initialization of the system and its environment (consisting of one administrator identified here by `bill`) and indicating that the number of communication company actor instances for the system's environment is 1 (identified here by `tango`).

2 the `actAdministrator` actor `bill` sends the message `oeLogin(icrashadmin,7WXC1359)` to securely connect to the system.

3 the `actAdministrator` actor `bill` sends the message `oeAddUser(1,Steve, pwdMessirExcalibur2017,1635242A75, 'Vehicular, Pedestrian, Wildlife, Property, Injury',Coordinator)` to set up a coordinator (i.e. identified here by `steve`) and indicating his identification information, ID (i.e. `1`) and a password (i.e. `pwdMessirExcalibur2017`), id from a token(i.e.`1635242A75`), specifying his domain of expertise(i.e. `'Vehicular, Pedestrian, Wildlife, Property, Injury'`) and finally he designates the user as a Coordinator(i.e.`Cooridnator`).

4 the `actAdministrator` actor `bill` sends the message `oeAddUser(2, franklin, pwdMessirExcalibur, 1456872B82, Null, DomainExpert)` create a domain expert who is one person(i.e. identified here by `franklin`) and indicating his user identification his identification, ID(i.e.`2`), password(i.e.`pwdMessirExcalibur123`), id from a token(i.e.`14566872B82`), specifying the users domain of expertise as(`Null` meaning that the user has no specific domain and can therefore only be an EMS user or Domain expert) and finally he designates the user as a domain expert(i.e.`DomainExpert`).

5 the `actAdministrator` actor `bill` sends the message `oeAddUser(3, barry, pwdEMSExcalibur321, 2436787C82, Null, EMS)` to set up a EMS user who is any user in the EMS headquarters(i.e. identified here by `barry`) and indicating his user identification as ID(i.e.`3`), password (i.e.`pwdEMSExcalibur421`), id from a token(i.e.`2436787C82`), specifying the users domain of expertise as(i.e.`Null` meaning that the user has no specific domain and can therefore only be an EMS user or DomainExpert) and finally he designates the user as an EMS user(i.e.`EMS`).

6 the `actAdministrator` actor `bill` sends the message `oeLogout()` to disconnect from the system.

7 the `actComCompany` actor `tango` sends the message `oeAlert(witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590, '3 cars involved in an accident.')` to transfer a declaration of a car crash by a witness indicating specific phone number, the date and time, the GPS coordinates of the witnessed car crash and a short message giving additional details.

8 the `actDomainExpert`actor `franklin` sends the message `oeLogin(franklin, pwdMessirExcalibur123, 687594FAD9)`to securely connect to the system entering his login(i.e.`franklin`), his password(i.e.`pwdMessirExcalibur123`)and entering his serial key that he reads form a token device given to him by the administrator(i.e.687594FAD9).

9 the `actEMS`actor `barry` sends the message `oeLogin(barry, pwdEMSExcalibur321, 24367872C282)`to securely connect to the system entering his login(i.e.`barry`), his password(i.e.`pwdEMSExcalibur321`)and entering his serial key that he reads form a token device given to him by the administrator(i.e.`24367872C282`).

10 the `actDomainExpert` actor `franklin` sends the message `oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')`to validate the crisis and to set a Domain to it.

11 the `actDomainExpert` actor `franklin` sends the message `oeSollicitateCrisisHandling()` indicating that there is a declared alert that is still not handled by any coordinator.

continues in next page . . .

. . . Use-Case Instance table continuation

12 the actCoordinator actor steve sends the message oeLogin(steve, pwdMessirExcalibur2017, 63524275D9) to securely connect to the system, entering his login(i.e.steve), his password(i.e.pwdMessirExcalibur2017) and his serial Key that he reads form a token device he is given by the administrator(i.e.63524275D9)

13 the actCoordinator actor steve sends the message oeGetAlertsSet(valid) to receive information about all the pending crisis.

14 the actCoordinator actor steve sends the message oeSetCrisisHandler(1,Medium, in-handling, 49.627095-6.160251, 2017-11-26-at-10-10-16AM) to declare that he is taking care of the alert with the ID equal to 1 which becomes the Crisis Id, sets the crisis type to(i.e.Medium, the crisis type to(i.e.in-handling), enters the GPS coordinates and the date and time.

15 the actComCompany actor tango sends the message oeAlert(witness, 2017-11-26-at-10-20-18AM,+3524666445314, 49.627095-6.160251,Please send rescue services.) to transfer a declaration of a car crash by a witness indicating specific the phone number, the date and time, the GPS coordinates of the witnessed car crash and a short message giving additional details. This alert's GPS coordinates match the previous alert sent coordinates in step 7 and the time elapsed between the two alerts is 10 minutes therefore the alert is considered to be originating from the same location.

16 the actDomainExpert actor franklin sends the message oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')to validate the crisis and to set a Domain to it. Because the alert originated from the same location and only 10 min later it an automated message informs the Domain Expert to validate it to the same alert ID as the previous alert.

17 the actCoordinator actor steve send the message oeReportOnCrisis(1, ' 2 Alerts received about a car accident at the same location apparently involving 3 vehicles', 2, Medium, Handled,'witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590,'3 cars involved in an accident', witness, 2017-11-26-at-10-2-18AM,+3524666445314, 49.627095-6.160251,'Please send rescue services.',3, 3) indicating the crisis ID(i.e.1), entering a specific comment in the comment area(i.e.'2Alertsrecived about a car accident at the same location apparently involving 3 vehicles'), specifying his user ID(i.e.2), setting the CrisisType(i.e.Medium), indicating the current crisisStatus(handled), entering the previous received alerts, indicating the number of vehicles involved in the accident(i.e.3) and entering the number of victims(i.e.3) because he suspects that there may be 3 victims due to the amount of cars involved in the accident. Entering a preliminary report on the crisis with the information that he presumes are right.

18 the actCoordinator actor steve send the message oeRequestEMSAssistance('We have received an Alert of an accident involving 3 cars from 1 victim and 1 witness at the same location please send police and ambulance',1, 49.627095-6.160251, 3,3, Ambulance Police), entering a comment(i.e. ' We have received an Alert of an accident involving 3 cars from 1 victim and 1 witness at the same location please send Police assistance'), indicating the RequestID(i.e.1), entering the GPS location of the accident(i.e.49.627095-6.160251), informing them of the number of cars involved(i.e.3) and informing them of the presumed number of victims(i.e.3) requesting emergency services assistance(i.e.Ambulance, Police).

19 the actCooordinator actor steve sends the message oeSetCrisisStatus(1, handled)to change the status of the crisis identified by the ID(i.e.1) to the status(i.e.handled) thus indicating that he has handled the situation for now but it's not solved yet.

20 the actEMS actor barry sends the message oeReplyToRequest(1,'Message received, dispatching police and ambulance units.', 1, Handled) indicating the CrisisID(i.e.1), sending the comment(i.e.'Message received, dispatching police and ambulance units.'), to confirm that the request with the ID(i.e.1) has been received and is being handled, indicated by the EMS crisis status(i.e. Handled).

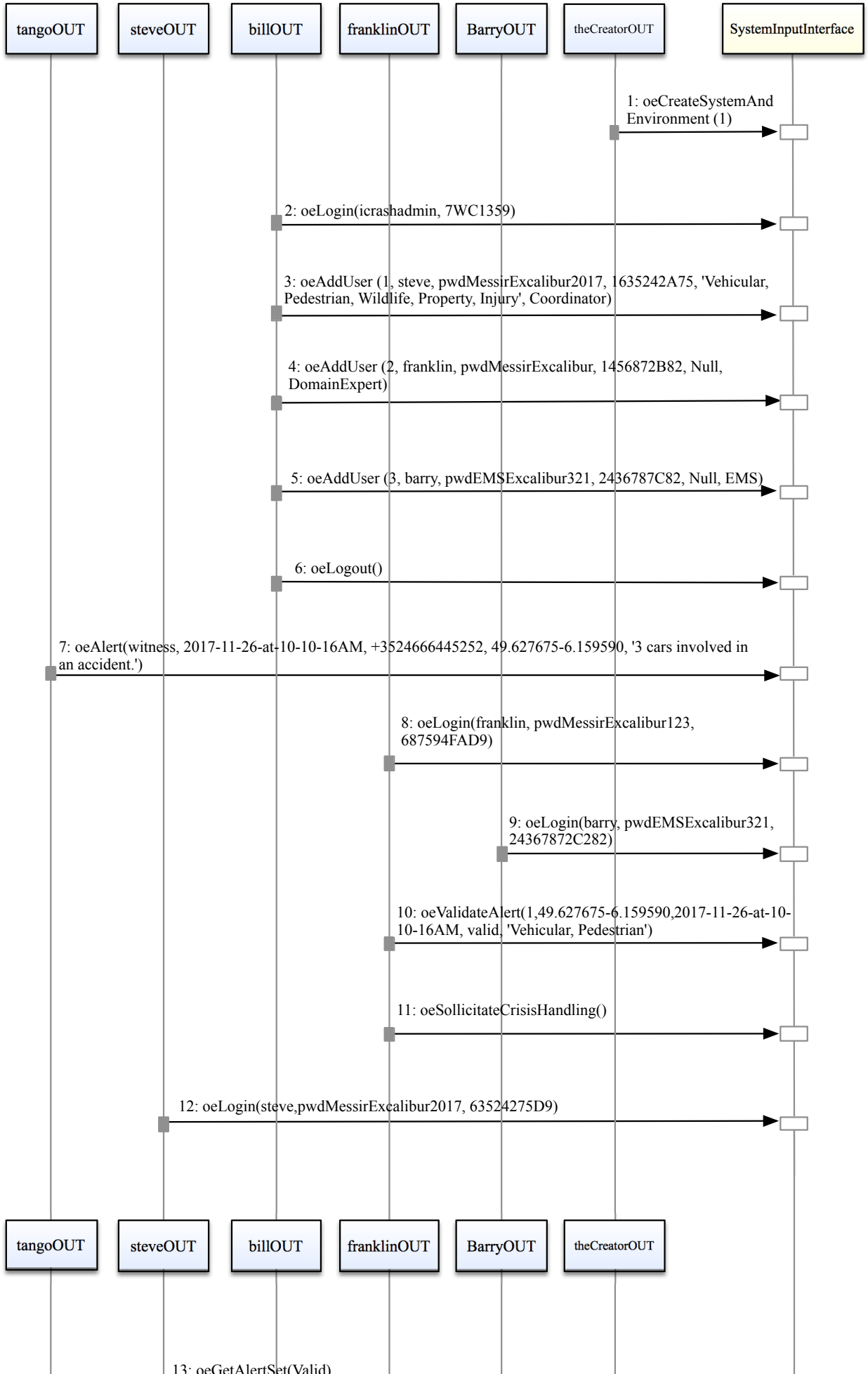
continues in next page . . .

. . . Use-Case Instance table continuation

21	the actEMS actor barry sends the message oeReportEMSCrisisStatus(1, 'Units arrived on scene report 4 victims, 3 cars involved in accident. Situation under control 1 victim brought Mercy hospital', solved, Mercy Hospital, Jeremy Springer, John Snow, +32168432) indicating the crisis ID(i.e.1), adding a comment(i.e. 'Units arrived on scene report 4 victims, 3 cars involved in accident. Situation under control 1 brought to Mercy hospital'), specifying the EMS crisis status as solved (i.e.solved), indicating the hospital name as (i.e.Mercy Hospital), indicating the victim name(i.e.Jeremy Springer), indicating the victims ICE contact's name as(i.e.John Snow) and the contacts phone number as (+32168432) to report on the EMS crisis Status.
22	the actComCompany actor tango sends message oeInfoFam('Dear John Snow, I regret to inform you that Jeremy Springer was in a car accident involving 3 other cars. Jeremy Springer was brought to the Mercy hospital for examination. Regretfully yours..',Jeremy Springer, +32168432, Mercy Hospital) to inform the victims family members about the state of the victim and in which hospital they reside.
23	the actCoordinator actor steve sends the message oeReportOnCrisis(1, ' 2 Alerts received about a car accident at the same location involving 3 vehicles and 4 victims one victim Jeremy Springer was brought to the Mercy hospital.', 2, Medium, Handled,'witness, 2017-11-26-at-10-10-16AM,+3524666445252, 49.627675-6.159590,'3 cars involved in an accident 'witness, 2017-11-26-at-10-2-18AM,+3524666445314, 49.627095-6.160251,'Please send rescue services.",3, 4) to set the report for the crisis with ID equal to 1 that he is handling.
24	the actCoordinator actor steve sends the message oeReportOnCrisis(1,'3 victims sent to hospital, 2 cars evacuated and 4 rescue unit mobilized') to set the report for the crisis with ID equal to 1 that he is handling.
25	the actCoordinator actor steve sends the message oeCloseCrisis(1) to declare the crisis with ID equal to 1 as closed.

Outputs

at the end of this instance flow the system should have its environment made of one communication company actor instance, one coordinator actor instance, one administrator instance and the creator instance. The system state should contain two alerts defined according to the information received from the communication company, only one crisis that should be considered as solved and being alerted by the two alerts received.


tangoOUT
steveOUT
billOUT
franklinOUT
BarryOUT
theCreatorOUT
SystemInputInterface
1: oeCreateSystemAnd Environment (1)
2: oeLogin(icrashadmin, 7WC1359)
3: oeAddUser (1, steve, pwdMessirExcalibur2017, 1635242A75, 'Vehicular, Pedestrian, Wildlife, Property, Injury', Coordinator)
4: oeAddUser (2, franklin, pwdMessirExcalibur, 1456872B82, Null, DomainExpert)
5: oeAddUser (3, barry, pwdEMSExcalibur321, 2436787C82, Null, EMS)
6: oeLogout()
7: oeAlert(witness, 2017-11-26-at-10-10-16AM, +3524666445252, 49.627675-6.159590, '3 cars involved in an accident.')
8: oeLogin(franklin, pwdMessirExcalibur123, 687594FAD9)
9: oeLogin(barry, pwdEMSExcalibur321, 24367872C282)
10: oeValidateAlert(1,49.627675-6.159590,2017-11-26-at-10-10-16AM, valid, 'Vehicular, Pedestrian')
11: oeSollicitateCrisisHandling()
12: oeLogin(steve,pwdMessirExcalibur2017, 63524275D9)
tangoOUT
steveOUT
billOUT
franklinOUT
BarryOUT
theCreatorOUT

Appendix B
Messir Specification Files Listing

B.1 File /concepts/primarytypes-associations.msr

```
1  package icrash.concepts.primarytypes.associations {
2
3  import icrash.concepts.primarytypes.datatypes
4  import icrash.concepts.primarytypes.classes
5  import icrash.environment
6
7  Concept Model {
8      Primary Types {
9
10 // Internal
11
12 association assctAlertctCrisis
13     ctAlert(rnAlerts)[1..*]
14     ctCrisis(rnTheCrisis)[1..1]
15
16 association assctAlertctHuman
17     ctAlert(rnSignaled)[1..*]
18     ctHuman (rnSignaler)[1..*]
19
20 association assctCrisisctCoordinator
21     ctCrisis(rnHandled)[0..*]
22     ctCoordinator(rnHandler)[0..1]
23
24 association assctCrisisctDomains
25     ctCrisis(rnTheCrisis)[1..1]
26     ctDomain(rnCrisisDomains)[1..1]
27
28 association assctCoordinatorctDomains
29     ctCoordinator(rnHandler)[1..1]
30     ctDomain(rnCrisisDomains)[1..1]
31
32 association assctAlertctDomains
33     ctAlert(rnAlerts)[1..1]
34     ctDomain(rnCrisisDomains)[1..1]
35
36 association assctAuthenticatedctToken
37     ctAuthenticated(rnctAuthenticated)[1..1]
38     ctToken(rnToken)[1..1]
39
40 // With Actors
41
42 association assctHumanactComCompany
43     ctHuman(rnctHuman)[0..*]
44     actComCompany(rnactComCompany)[1..1]
45
46 association assctCoordinatoractCoordinator
47     ctCoordinator(rnctCoordinator)[1..1]
48     actCoordinator(rnactCoordinator)[1..1]
49
50 association assctAuthenticatedactAuthenticated
51     ctAuthenticated(rnctAuthenticated)[1..1]
52     actAuthenticated(rnctactAuthenticated)[1..1]
53
54 association assctDomainExpertDomainExpert
55     ctDomainExpert(rnctDomainExpert)[1..1]
56     actDomainExpert(rnactDomainExpert)[1..1]
57
58 }}}
```

Listing B.1 Messir Spec. file primarytypes-associations.msr.

B.2 File /concepts/primarytypes-classes.msr

```
1  package icrash.concepts.primarytypes.classes {
2
3  import icrash.concepts.primarytypes.datatypes
4  import icrash.environment
5  import lu.uni.lassy.messir.libraries.primitives
6  import lu.uni.lassy.messir.libraries.math
7  import lu.uni.lassy.messir.libraries.calendar
8
9  Concept Model {
10     Primary Types {
11         state class ctState {
12             attribute nextValueForAlertID:dtInteger
13             attribute nextValueForCrisisID:dtInteger
14             attribute clock:dtDateAndTime
15             attribute crisisReminderPeriod:dtSecond
16             attribute initialCrisisReminderPeriod:dtSecond
17             attribute maxCrisisReminderPeriod:dtSecond
18             attribute vpStarted:ptBoolean
19             operation init(
20                 dtInteger AnextValueForAlertID,
21                 dtInteger AnextValueForCrisisID,
22                 dtDateAndTime Aclock,
23                 dtSecond AcrisisReminderPeriod,
24                 dtSecond AinitialCrisisReminderPeriod,
25                 dtSecond AmaxCrisisReminderPeriod,
26                 ptBoolean AvpStarted): ptBoolean
27         }
28
29         class ctAlert role rnctAlert cardinality [0..*] {
30             attribute id:dtAlertID
31             attribute status: etAlertStatus
32             attribute location:dtGPSLocation
33             attribute instant:dtDateAndTime
34             attribute comment:dtComment
35             operation init(
36                 dtAlertID Aid,
37                 etAlertStatus Astatus,
38                 dtGPSLocation Alocation,
39                 dtDateAndTime Ainstant,
40                 dtComment Acomment): ptBoolean
41         }
42
43         class ctCrisis role rnctCrisis cardinality [0..*] {
44             attribute id:dtCrisisID
45             attribute type:etCrisisType
46             attribute status: etCrisisStatus
47             attribute location:dtGPSLocation
48             attribute instant:dtDateAndTime
49             attribute comment:dtComment
50             operation init(
51                 dtCrisisID Aid,
52                 etCrisisType Atype,
53                 etCrisisStatus Astatus,
54                 dtGPSLocation Alocation,
55                 dtDateAndTime Ainstant): ptBoolean
56         }
57
58         class ctHuman role rnctHuman cardinality [0..*] {
59             attribute id:dtPhoneNumber
60             attribute name:dtName
61             attribute kind:etHumanKind
62             operation init(
63                 dtName AName,
64                 dtPhoneNumber Aid,
65                 etHumanKind Akind): ptBoolean
66         }
67
68         class ctAuthenticated role rnctAuthenticated cardinality [0..*] {
69             attribute login:dtLogin
70             attribute pwd: dtPassword
71             attribute toeknID: dtTokenID
72             attribute tokenkey: dtKey
73             attribute vpIsLogged: ptBoolean
74             operation init(
75                 dtLogin Alogin,
76                 dtPassword Apwd,
77                 dtTokenID Atokenid): ptBoolean
```

```
78          }
79
80          class ctCoordinator role rnctCoordinator cardinality [0..*] extends ctAuthenticated {
81              attribute id: dtUserID
82              operation init(
83                  dtUserID Aid,
84                  dtLogin Alogin,
85                  dtPassword Apwd,
86                  dtTokenID Atokenid): ptBoolean
87          }
88
89          class ctAdministrator role rnctAdministrator cardinality [1..1] extends ctAuthenticated {
90              operation init(
91                  dtLogin Alogin,
92                  dtPassword Apwd,
93                  dtTokenID Atokenid): ptBoolean
94          }
95
96          class ctDomain role rnctDomains cardinality [1..*] {
97              attribute vpIsPedestrian: ptBoolean
98              attribute vpIsVehicular: ptBoolean
99              attribute vpIsWildlife: ptBoolean
100             attribute vpIsProperty: ptBoolean
101             attribute vpIsInjury: ptBoolean
102             operation init(
103                 ptBoolean AvpIsPedestrian,
104                 ptBoolean AvpIsVehicular,
105                 ptBoolean AvpIsWildlife,
106                 ptBoolean AvpIsProperty,
107                 ptBoolean AvpIsInjury): ptBoolean
108         }
109
110         class ctEMStype role rnctEMStypesrequested cardinality [1..*] {
111             attribute vpRequestFireFigher: ptBoolean
112             attribute vpRequestPolice: ptBoolean
113             attribute vpRequestAmbulance: ptBoolean
114             operation init(
115                 ptBoolean AvpRequestFireFighter,
116                 ptBoolean AvpRequestPolice,
117                 ptBoolean AvpRequestAmublance): ptBoolean
118         }
119
120         class ctVictim role victimofaCarCrash cardinality [1..*] extends ctHuman {
121             attribute VictimICEPhoneNumber: dtPhoneNumber
122             attribute VictimICEContactName: dtName
123             operation init(
124                 ptBoolean VictimICEPhoneNumber,
125                 ptBoolean VictimICEContactName): ptBoolean
126         }
127
128         class ctToken role rnctSysmetricKey cardinality [1..*] {
129             attribute symetricKey: dtKey
130             attribute tokenID: dtTokenID
131             operation init(
132                 dtTokenID AsymetricKey,
133                 dtKey Akey): ptBoolean
134         }
135
136         class ctDomainExpert role rnctDomainExpert cardinality [1..*] extends ctAuthenticated {
137             attribute id: dtUserID
138             operation init(
139                 dtUserID Aid,
140                 dtLogin Alogin,
141                 dtPassword Apwd,
142                 dtTokenID Atokenid): ptBoolean
143         }
144 }}}
```

Listing B.2 Messir Spec. file primarytypes-classes.msr.

B.3 File /concepts/primarytypes-datatypes.msr

```
1 package icrash.concepts.primarytypes.datatypes {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.string
5 import lu.uni.lassy.messir.libraries.math
6 import lu.uni.lassy.messir.libraries.calendar
7
8 Concept Model {
9     Primary Types {
```

```
10          datatype dtAlertID extends dtString {
11              operation is(): ptBoolean
12          }
13
14          datatype dtCrisisID extends dtString {
15              operation is(): ptBoolean
16              operation isSameCrisis(
17                  dtDate aDate,
18                  dtTime aTimeandDate,
19                  dtDate bDate,
20                  dtTime bTime): ptBoolean
21          }
22
23          datatype dtLogin extends dtString {
24              operation is(): ptBoolean
25          }
26
27          datatype dtPassword extends dtString {
28              operation is(): ptBoolean
29          }
30
31          datatype dtUserID extends dtString {
32              operation is(): ptBoolean
33          }
34
35          datatype dtHumanName extends dtString {
36              operation is(): ptBoolean
37          }
38
39          datatype dtPhoneNumber extends dtString {
40              operation is(): ptBoolean
41          }
42
43          datatype dtComment extends dtString {
44              operation is(): ptBoolean
45          }
46
47          datatype dtLatitude extends dtReal {
48              operation is(): ptBoolean
49          }
50
51          datatype dtLongitude extends dtReal {
52              operation is(): ptBoolean
53          }
54
55          datatype dtKey extends dtString {
56              operation is(): ptBoolean
57          }
58
59          datatype dtTokenID extends dtString {
60              operation is(): ptBoolean
61          }
62
63          datatype dtGPSLocation {
64              attribute latitude: dtLatitude
65              attribute longitude: dtLongitude
66              operation is(): ptBoolean
67              operation isNearTo(
68                  dtGPSLocation aGPSLocation,
69                  dtGPSLocation bGPSLocation): ptBoolean
70          }
71
72          datatype dtRecievedMessage extends dtString {
73              operation is(): ptBoolean
74          }
75
76          datatype dtHospitalName extends dtString {
77              operation is(): ptBoolean
78          }
79
80          datatype dtNbrofVehiculesInAccident extends dtReal {
81              operation is(): ptBoolean
82          }
83
84          datatype dtNbrOfVictims extends dtReal {
85              operation is(): ptBoolean
86          }
87
88          datatype dtRequestID extends dtString {
89              operation is(): ptBoolean
90          }
91
92          datatype dtName extends dtString {
93              operation is(): ptBoolean
```

```
94          }
95
96          datatype dtVictimICEContactName extends dtString {
97              operation is(): ptBoolean
98          }
99
100         datatype dtVictimICEContactPhoneNumber extends dtString {
101             operation is(): ptBoolean
102         }
103
104         enum etEMSCrisisStatus {
105             constants["pending", "handled", "solved"]
106             operation is(): ptBoolean
107         }
108
109         enum etCrisisStatus {
110             constants["in-handling", "handled", "solved"]
111             operation is(): ptBoolean
112         }
113
114         enum etAlertStatus {
115             constants["pending", "valid", "invalid"]
116             operation is(): ptBoolean
117         }
118
119         enum etCrisisType {
120             constants["Small","Medium","Huge"]
121             operation is(): ptBoolean
122         }
123
124         enum etHumanKind {
125             constants["witness", "victim", "anonymous"]
126             operation is(): ptBoolean
127         }
128
129         enum etUserType {
130             constants["Coordinator","DomainExpert","EMS"]
131             operation is():ptBoolean
132         }
133
134 }}}
```

Listing B.3 Messir Spec. file primarytypes-datatypes.msr.

B.4 File /concepts/secondarytypes-associations.msr

```
1 package icrash.concepts.secondarytypes.associations {
2     Concept Model {
3         Secondary Types{
4 }}}
```

Listing B.4 Messir Spec. file secondarytypes-associations.msr.

B.5 File /concepts/secondarytypes-classes.msr

```
1 package icrash.concepts.secondarytypes.classes {
2     Concept Model {
3         Secondary Types {
4 }}}
```

Listing B.5 Messir Spec. file secondarytypes-classes.msr.

B.6 File /concepts/secondarytypes-datatypes.msr

```
1  package icrash.concepts.secondarytypes.datatypes {
2
3  import lu.uni.lassy.messir.libraries.primitives
4  import lu.uni.lassy.messir.libraries.string
5  import icrash.concepts.primarytypes.datatypes
6
7  Concept Model {
8      Secondary Types {
9          datatype dtSMS {
10             attribute value: ptString
11             operation is():ptBoolean
12     }
```

```
}}}
```

Listing B.6 Messir Spec. file secondarytypes-datatypes.msr.

B.7 File /environment/environment.msr

```
package icrash.environment {
import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.math
import lu.uni.lassy.messir.libraries.calendar

Environment Model {
    actor actMsrCreator role rnactMsrCreator cardinality [0..1] {
        input interface inactMsrCreator {
        }

        output interface outactMsrCreator {
            operation oeCreateSystemAndEnvironment(ptInteger AqtyComCompanies): ptBoolean
        }
    }

    actor actAdministrator role rnactAdministrator cardinality [1..1] extends actAuthenticated {
        operation init(): ptBoolean
        output interface outactAdministrator {
            operation oeAddUser(
                dtUserID AdtUserID,
                dtLogin AdtLogin,
                dtPassword AdtPassword,
                dtTokenID AdtTokenID,
                etUserType AdtUserType,
                ctDomain ActDomain): ptBoolean

                operation oeDeleteUser(dtUserID AdtUserID): ptBoolean
        }

        input interface inactAdministrator {
            operation ieUserAdded(): ptBoolean
            operation ieUserDeleted(): ptBoolean
        }
    }

    actor actCoordinator role rnctactCoordinator cardinality [0..*] extends actAuthenticated {
        operation init():ptBoolean
        output interface outactCoordinator {
            operation oeCloseAlert(dtAlertID AdtAlertID): ptBoolean
            operation oeCloseCrisis(dtCrisisID AdtCrisisID): ptBoolean
            operation oeGetAlertsSet(etAlertStatus AetAlertStatus): ptBoolean
            operation oeGetCrisisSet(etCrisisStatus AetCrisisStatus): ptBoolean
            operation oeSetCrisisHandler(dtCrisisID AdtCrisisID,
                etCrisisType AetCrisisType,
                etCrisisStatus AetCrisisStatus,
                dtGPSLocation AdtGPSLocatino,
                dtDateAndTime AdtDateAndTime): ptBoolean
            operation oeReportOnCrisis(dtCrisisID AdtCrisisID,
                dtComment AdtComment, dtUserID AdtUserID,
                etCrisisType AdtCrisisType,
                etCrisisStatus AdtCrisisStatus,
                dtRecievedMessage AdtReceivedMessage,
                dtNbrofVehiculesInAccident AdtVehiculesInAccident,
                dtNbrOfVictims AdtNbrVictims): ptBoolean
            operation oeSetCrisisStatus(dtCrisisID AdtCrisisID,
                etCrisisStatus AetCrisisStatus): ptBoolean
            operation oeRequestEMSAssistance(dtComment AdtComment,
                dtRequestID AdtRequest,
                dtGPSLocation AdtGPSLocation,
                dtNbrofVehiculesInAccident AdtVehiculesInAccident,
                dtNbrOfVictims AdtNbrVictims,
                ctEMStype ActEMSType): ptBoolean
        }

        input interface inactCoordinator {
            operation ieSendAnAlert(ctAlert AclAlert): ptBoolean
            operation ieSendACrisis(ctCrisis ActCrisis): ptBoolean
            operation ieMessage(ptString AMessage): ptBoolean
        }
    }

    actor actComCompany role rnactComCompany cardinality [0..*] {
```

```
76          operation init(): ptBoolean
77          output interface outactComCompany {
78              operation oeAlert(etHumanKind AetKind,
79                  dtDate AdtMyDate,
80                  dtTime AdtTime,
81                  dtPhoneNumber AdtPhoneNumber,
82                  dtGPSLocation AdtGPSLocation,
83                  dtComment AdtComment): ptBoolean
84              operation oeInfoFam(ctVictim ActVictim,
85                  dtHospitalName AHospitalName): ptBoolean
86          }
87
88          input interface inactComCompany {
89              operation ieSmsSend(dtPhoneNumber AdtPhoneNumber,
90                  dtSMS AdtSMS): ptBoolean
91          }
92      }
93
94      actor actAuthenticated role rnactAuthenticated cardinality [0..*] {
95          operation init(): ptBoolean
96          output interface outactAuthenticated {
97              operation oeLogin(dtLogin AdtLogin,
98                  dtPassword AdtPassword,
99                  dtKey Adtkey): ptBoolean
100             operation oeLogout(): ptBoolean
101         }
102
103         input interface inactAuthenticated {
104             operation ieMessage(ptString AMessage): ptBoolean
105         }
106     }
107
108     actor actDomainExpert role rnactDomainExpert cardinality [1..*] {
109         operation init(): ptBoolean
110         input interface inDomainExpert {
111             operation ieAlertRecieved(): ptBoolean
112         }
113
114         output interface outactDomainExpert {
115             operation oeValidateAlert(dtAlertID AdtAlertID,
116                 dtGPSLocation AdtGPSLocation,
117                 dtDateAndTime AdtDateandTime,
118                 etAlertStatus edtAlertStatus,
119                 ctDomain ActDomain): ptBoolean
120             operation oeSollicitateCrisisHandling(): ptBoolean
121         }
122     }
123
124     actor actEMS role rnactEMS cardinality [1..*] {
125         operation init(): ptBoolean
126         input interface inactEMS {
127             operation ieRequest(): ptBoolean
128         }
129
130         output interface outactEMS {
131             operation oeReportEMSCrisisStatus(dtCrisisID AdtCrisisID,
132                 dtComment AdtComment,
133                 etEMSCrisisStatus AdtEMSCrisisStatus,
134                 dtHospitalName AdtHospitalName,
135                 ctVictim ActVictim): ptBoolean
136             operation oeReplyToRequest(dtCrisisID AdtCrisisID,
137                 dtComment AdtComment,
138                 dtRequestID AdtRequest,
139                 etEMSCrisisStatus AdtEMsCrisisStatus): ptBoolean
140         }
141     }
142 }}
```

Listing B.7 Messir Spec. file environment.msr.

B.8 File /operations/concepts/primarytypes-classes/primarytypes-classes-ctAdministrator.msr

```
1 /*
2 * @author nicolas.guelfi
3 * @date Mon Sep 30 17:11:03 CEST 2013
4 */
5
6 package icrash.operations.concepts.primarytypes.classes.ctAdministrator{
7
8 import lu.uni.lassy.messir.libraries.primitives
```

```
9  import lu.uni.lassy.messir.libraries.calendar
10 import lu.uni.lassy.messir.libraries.math
11
12 import icrash.concepts.primarytypes.datatypes
13 import icrash.concepts.primarytypes.classes
14 import icrash.concepts.secondarytypes.datatypes
15 import icrash.concepts.secondarytypes.classes
16
17 Operation Model {
18
19   operation: icrash.concepts.primarytypes.classes.ctAdministrator.init(
20       dtLogin Alogin,
21       dtPassword Apwd,
22       dtTokenID Awptoken
23   ):ptBoolean
24     preF: true
25     postF: true
26
27   }
28 }
```

Listing B.8 Messir Spec. file primarytypes-classes-ctAdministrator.msr.

B.9 File /operations/concepts/primarytypes-classes/primarytypes-classes-ctAlert.msr

```
1  /*
2  * @author nicolas.guelfi
3  * @date Mon Sep 30 17:11:03 CEST 2013
4  */
5
6  package icrash.operations.concepts.primarytypes.classes.ctAlert{
7
8  import lu.uni.lassy.messir.libraries.primitives
9  import lu.uni.lassy.messir.libraries.calendar
10
11 import icrash.concepts.primarytypes.datatypes
12 import icrash.concepts.primarytypes.classes
13 import icrash.concepts.secondarytypes.datatypes
14 import icrash.concepts.secondarytypes.classes
15
16 Operation Model {
17
18 operation: icrash.concepts.primarytypes.classes.ctAlert.init(dtAlertID Aid, etAlertStatus Astatus,
       dtGPSLocation Alocation, dtDateAndTime Ainstant, dtComment Acomment):ptBoolean
19   preF: true
20   postF: true
21
22   }
23 }
```

Listing B.9 Messir Spec. file primarytypes-classes-ctAlert.msr.

B.10 File /operations/concepts/primarytypes-classes/primarytypes-classes-ctState.msr

```
1  /*
2  * @author nicolas.guelfi
3  * @date Mon Sep 30 17:11:03 CEST 2013
4  */
5
6  package icrash.operations.concepts.primarytypes.classes.ctState{
7
8  import lu.uni.lassy.messir.libraries.primitives
9  import lu.uni.lassy.messir.libraries.calendar
10 import lu.uni.lassy.messir.libraries.math
11
12 import icrash.concepts.primarytypes.datatypes
13 import icrash.concepts.primarytypes.classes
14 import icrash.concepts.secondarytypes.datatypes
15 import icrash.concepts.secondarytypes.classes
16
17 Operation Model {
18
19   operation: icrash.concepts.primarytypes.classes.ctState.init(
20     dtInteger AnextValueForAlertID,
21     dtInteger AnextValueForCrisisID,
22     dtDateAndTime Aclock,
```

```
23      dtSecond AcrisisReminderPeriod,
24      dtSecond AinitialCrisisReminderPeriod,
25      dtSecond AmaxCrisisReminderPeriod,
26      ptBoolean AvpStarted
27    ):ptBoolean
28      preF: true
29      postF: true
30
31    }
32  }
```

Listing B.10 Messir Spec. file primarytypes-classes-ctState.msr.

B.11 File /operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtAlertID.msr

```
1  /*
2  * @author nicolas.guelfi
3  * @date Mon Sep 30 17:11:03 CEST 2013
4  */
5
6  package icrash.operations.concepts.primarytypes.datatypes.dtAlertID{
7
8  import lu.uni.lassy.messir.libraries.primitives
9
10 import icrash.concepts.primarytypes.datatypes
11 import icrash.concepts.primarytypes.classes
12 import icrash.concepts.secondarytypes.datatypes
13 import icrash.concepts.secondarytypes.classes
14
15 Operation Model {
16
17     operation: icrash.concepts.primarytypes.datatypes.dtAlertID.is():ptBoolean
18     ocl:"
19 context PrimaryTypesDatatypes::dtAlertID::
20 is():Boolean
21 /*-------------------------------------------*/
22 pre:
23 /* Pre Functional:*/
24 /* Pre F01 */
25 true
26 post:
27 let TheResult:Boolean in
28 if
29 (
30 /* Post Functional:*/
31 /* Post F01 */
32 self.value > 0
33 )
34 then TheResult = true
35 else TheResult = false
36 endif
37 and result = TheResult
38
39 /*-------------------------------------------*/
40 "
41   }
42 }
```

Listing B.11 Messir Spec. file primarytypes-datatypes-dtAlertID.msr.

B.12 File /operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtComment.msr

```
1  /*
2  * @author nicolas.guelfi
3  * @date Mon Sep 30 17:11:03 CEST 2013
4  */
5
6  package icrash.operations.concepts.primarytypes.datatypes.dtComment{
7
8  import lu.uni.lassy.messir.libraries.primitives
9
10 import icrash.concepts.primarytypes.datatypes
11 import icrash.concepts.primarytypes.classes
12 import icrash.concepts.secondarytypes.datatypes
13 import icrash.concepts.secondarytypes.classes
14
```

```
Operation Model {

    operation: icrash.concepts.primarytypes.datatypes.dtComment.is():ptBoolean
    ocl:"
context PrimaryTypesDatatypes::dtComment::
is():Boolean
/*-------------------------------------------*/
pre:
/* Pre Functional:*/
/* Pre F01 */
true
post:
let TheResult:Boolean in
let MaxLength:Integer in
if
(
/* Post Functional:*/
/* Post F01 */
MaxLength = 160
and self.value.size() <= MaxLength
)
then TheResult = true
else TheResult = false
endif
and result = TheResult
/*-------------------------------------------*/
"
  }
}
```

Listing B.12 Messir Spec. file primarytypes-datatypes-dtComment.msr.

B.13 File /operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtPhoneNumber.msr

```
/*
* @author nicolas.guelfi
* @date Mon Sep 30 17:11:03 CEST 2013
*/

package icrash.operations.concepts.primarytypes.datatypes.dtPhoneNumber{

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes

Operation Model {

    operation: icrash.concepts.primarytypes.datatypes.dtPhoneNumber.is():ptBoolean
    ocl:"
context PrimaryTypesDatatypes::dtPhoneNumber::
is():Boolean
/*-------------------------------------------*/
pre:
/* Pre Functional:*/
/* Pre F01 */
true
post:
let TheResult:Boolean in
if
(
/* Post Functional:*/
/* Post F01 */
self.value.size() = 10
)
then TheResult = true
else TheResult = false
endif
and result = TheResult
/*-------------------------------------------*/
"
  }
}
```

Listing B.13 Messir Spec. file primarytypes-datatypes-dtPhoneNumber.msr.

B.14 File /operations/concepts/secondarytypes-datatypes/dtSMS.msr

```
/*
* @author nicolas.guelfi
* @date Mon Sep 30 17:11:03 CEST 2013
*/

package icrash.operations.concepts.secondarytypes.datatypes.dtSMS{

import lu.uni.lassy.messir.libraries.primitives
import lu.uni.lassy.messir.libraries.calendar
import lu.uni.lassy.messir.libraries.math

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes

Operation Model {

operation: icrash.concepts.secondarytypes.datatypes.dtSMS.is():ptBoolean
  preF: true
  postF: true

  }
}
```

Listing B.14 Messir Spec. file dtSMS.msr.

B.15 File /operations/environment/environment-actAdministrator.msr

```
package icrash.operations.environment.actAdministrator{

import icrash.environment

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
import icrash.concepts.secondarytypes.datatypes
import icrash.concepts.secondarytypes.classes

Operation Model {

operation: icrash.environment.actAdministrator.outactAdministrator.oeAddUser(
                        dtUserID AdtUseroID,
                        dtLogin AdtLogin,
                        dtPassword AdtPassword,
                        dtTokenID AdtTokenID,
                        etUserType AdtUserType,
                        ctDomain AdtDomain
                        ):ptBoolean
  preF: true
  postF: true
  ocl:""

operation: icrash.environment.actAdministrator.outactAdministrator.oeDeleteUser(
                        dtUserID AdtUserID):ptBoolean
  preF: true
  postF: true
  ocl:""

}
}
```

Listing B.15 Messir Spec. file environment-actAdministrator.msr.

B.16 File /operations/environment/environment-actAuthenticated.msr

```
package icrash.operations.environment.actAuthenticated{

import icrash.environment

import lu.uni.lassy.messir.libraries.primitives

import icrash.concepts.primarytypes.datatypes
import icrash.concepts.primarytypes.classes
```

```
9  import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11
12 Operation Model {
13
14 operation: icrash.environment.actAuthenticated.outactAuthenticated.oeLogin(dtLogin AdtLogin,
       dtPassword AdtPassword, dtKey Adtkey):ptBoolean
15 //----------------------------------------
16 preF: true
17 //----------------------------------------
18 preP:
19 // PreP 01
20 let theSystemPre:ctState in
21 theSystem
22
23 operation: icrash.environment.actAuthenticated.outactAuthenticated.oeLogout():ptBoolean
24   preF: true
25   postF: true
26
27 }
28 }
```

Listing B.16 Messir Spec. file environment-actAuthenticated.msr.

B.17 File /operations/environment/environment-actComCompany.msr

```
1  package icrash.operations.environment.actComCompany{
2
3  import icrash.environment
4
5  import lu.uni.lassy.messir.libraries.primitives
6  import lu.uni.lassy.messir.libraries.calendar
7  import lu.uni.lassy.messir.libraries.math
8
9  import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11 import icrash.concepts.secondarytypes.datatypes
12 import icrash.concepts.secondarytypes.classes
13
14 Operation Model {
15
16 operation: icrash.environment.actComCompany.outactComCompany.oeAlert(etHumanKind AetKind, dtDate
       AdtMyDate, dtTime AdtTime, dtPhoneNumber AdtPhoneNumber, dtGPSLocation AdtGPSLocation, dtComment
        AdtComment):ptBoolean
17   preF: true
18   preP: true
19   postF: true
20   postP: true
21   ocl: "iyyiyuuouhuhuhulhhu
22 hii
23 preF: if coordinator"
24
25   }
26 }
```

Listing B.17 Messir Spec. file environment-actComCompany.msr.

B.18 File /operations/environment/environment-actCoordinator.msr

```
1
2  /*
3  * @author Alexander
4  * @date Wed Feb 26 15:06:34 CET 2014
5  */
6  package lu.uni.lassy.excalibur.examples.icrash.operations.environment.actCoordinator {
7
8  import lu.uni.lassy.messir.libraries.primitives
9  import lu.uni.lassy.messir.libraries.calendar
10 import lu.uni.lassy.messir.libraries.math
11
12 import icrash.concepts.primarytypes.datatypes
13 import icrash.concepts.primarytypes.classes
14 import icrash.concepts.secondarytypes.datatypes
15 import icrash.concepts.secondarytypes.classes
16
17   Operation Model {
18
19   operation: icrash.environment.actCoordinator.outactCoordinator.oeCloseAlert(dtAlertID AdtAlertID):
       ptBoolean //never used
```

```
20      preF: true
21      postF: true
22      ocl: ""
23
24    operation: icrash.environment.actCoordinator.outactCoordinator.oeCloseCrisis(dtCrisisID
          AdtCrisisoID):ptBoolean
25      preF: true
26      postF: true
27
28    operation: icrash.environment.actCoordinator.outactCoordinator.oeGetAlertsSet(etAlertStatus
          AetAlertStatus):ptBoolean
29      preF: true
30      postF: true
31
32    operation: icrash.environment.actCoordinator.outactCoordinator.oeGetCrisisSet(etCrisisStatus
          AetCrisisoStatus):ptBoolean//Never used
33      preF: true
34      postF: true
35
36    operation: icrash.environment.actCoordinator.outactCoordinator.oeSetCrisisHandler(dtCrisisID
          AdtCrisisuID,
37                                                etCrisisType AetCrisisType,
38                                                etCrisisStatus AetCrisisuStatus,
39                                                dtGPSLocation AdtGPSLocatino,
40                                                dtDateAndTime AdtDateAndTime
41                                                ):ptBoolean
42      preF: true
43      postF: true
44
45    operation: icrash.environment.actCoordinator.outactCoordinator.oeReportOnCrisis(
46                          dtCrisisID AdtCrisislID,
47                          dtComment AdtCommento,
48                          dtUserID AdtUserID,
49                          etCrisisType AdtCrisisType,
50                          etCrisisStatus AdtCrisisStatus,
51                          dtRecievedMessage AdtReceivedMessage,
52                          dtNbrofVehiculesInAccident AdtVehiculesInAccidento,
53                          dtNbrOfVictims AdtNbrVictimso
54                          ):ptBoolean
55      preF: true
56      postF: true
57
58    operation: icrash.environment.actCoordinator.outactCoordinator.oeSetCrisisStatus(
59                          dtCrisisID AdtCrisisID,
60                          etCrisisStatus AetCrisisStatus
61                          ):ptBoolean
62      preF: true
63      postF: true
64
65    operation: icrash.environment.actCoordinator.outactCoordinator.oeRequestEMSAssistance(
66                          dtComment AdtComment,
67                          dtRequestID AdtRequest,
68                          dtGPSLocation AdtGPSLocation,
69                          dtNbrofVehiculesInAccident AdtVehiculesInAccident,
70                          dtNbrOfVictims AdtNbrVictims,
71                          ctEMStype ActEMSType
72                          ):ptBoolean
73      preF: true
74      postF: true
75
76  }
77
78  }
```

Listing B.18 Messir Spec. file environment-actCoordinator.msr.

B.19 File /operations/environment/environment-actDomainExpert.msr

```
1  package icrash.operations.environment.actDomainExpert{
2
3  import icrash.environment
4
5  import lu.uni.lassy.messir.libraries.primitives
6  import lu.uni.lassy.messir.libraries.calendar
7  import lu.uni.lassy.messir.libraries.math
8
9  import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11 import icrash.concepts.secondarytypes.datatypes
12 import icrash.concepts.secondarytypes.classes
13
```

```
14 Operation Model {
15
16 operation: icrash.environment.actDomainExpert.outactDomainExpert.oeValidateAlert(
17                         dtAlertID AdtAlertID,
18                         dtGPSLocation AdtGPSLocation,
19                         dtDateAndTime AdtDateandTime,
20                         etAlertStatus edtAlertStatus,
21                         ctDomain ActDomain
22                         ):ptBoolean
23   preF: true
24   postF: true
25
26 operation: icrash.environment.actDomainExpert.outactDomainExpert.oeSollicitateCrisisHandling() :
       ptBoolean
27   preF: true
28   postF: true
29
30   }
31 }
```

Listing B.19 Messir Spec. file environment-actDomainExpert.msr.

B.20 File /operations/environment/environment-actEMS.msr

```
1 /*
2 * @author Alexander
3 * @date Mon Mar 31 11:21:28 CEST 2014
4 */
5
6 package lu.uni.lassy.excalibur.examples.icrash.operations.environment.environment.actEMS {
7
8 import lu.uni.lassy.messir.libraries.primitives
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11 import icrash.concepts.secondarytypes.datatypes
12 import icrash.concepts.secondarytypes.classes
13
14   Operation Model {
15
16     operation: icrash.environment.actEMS.outactEMS.oeReportEMSCrisisStatus(dtCrisisID AdtCrisisIDo,
17                         dtComment AdtCommento,
18                         etEMSCrisisStatus AdtEMSCrisisStatus,
19                         dtHospitalName AdtHospitalName,
20                         ctVictim ActVictim
21                         ):ptBoolean
22       preF: true
23       postF: true
24
25     operation: icrash.environment.actEMS.outactEMS.oeReplyToRequest(
26                         dtCrisisID AdtCrisisID,
27                         dtComment AdtComment,
28                         dtRequestID AdtRequest,
29                         etEMSCrisisStatus AdtEMsCrisisStatus
30                         ):ptBoolean
31       preF: true
32       postF: true
33
34   }
35
36 }
```

Listing B.20 Messir Spec. file environment-actEMS.msr.

B.21 File /operations/environment/environment-actMsrCreator.msr

```
1 package icrash.operations.environment.actMsrCreator{
2
3 import icrash.environment
4
5 import lu.uni.lassy.messir.libraries.primitives
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11
12 Operation Model {
13
```

```
14 operation: icrash.environment.actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment(ptInteger
       AqtyComCompanies):ptBoolean
15
16 ocl:
17 "
18 context Environment::outactMsrCreator::
19 oeCreateSystemAndEnvironment(qtyComCompanies:Integer)
20 /*-------------------------------------------*/
21 pre:
22 /* Pre Protocol:*/
23 /* PreF01 */
24 true
25 /* Pre Functional:*/
26 /* PreP01 */
27 and true
28 /*-------------------------------------------*/
29 post:
30 let TheState:PrimaryTypesClasses::ctState in
31 let AactMsrCreator:Environment::actMsrCreator in
32
33 /* Post Functional:*/
34 /* PostF01 */
35 TheState._init(1,true)
36
37 /* PostF02 */
38 and TheState.rnactComCompany->size() = qtyComCompanies
39 and TheState.rnactComCompany
40   ->forAll(cca:Environment::actComCompany | cca._init())
41
42 /* PostF03 */
43 and AactMsrCreator._init()
44
45 /* Post Protocol:*/
46 /* PostP01 */
47 and TheState.vpmStarted = true
48 "
49
50     operation: icrash.environment.actComCompany.init():ptBoolean
51
52 ocl:
53 "
54 context Environment::actComCompany::_init():Boolean
55 /*-------------------------------------------*/
56 pre: true
57 /*-------------------------------------------*/
58 post:
59 let pIN:Environment::inactComCompany in
60 let pOUT:Environment::outactComCompany in
61 if (
62 /* Post Functional:*/
63 /* Post F01 */
64     self.oclIsNew()
65 /* Post F02 */
66     and pIN.oclIsNew()
67     and pOUT.oclIsNew()
68     and pIN = self.InterfaceIN
69     and pOUT= self.InterfaceOUT
70       )
71     then result = true
72     else result = false
73     endif
74  /*-------------------------------------------*/"
75 }
76 }
```

Listing B.21 Messir Spec. file environment-actMsrCreator.msr.

References

1. Guelfi, N.: Messir: A Scientific Methods for the Sofware Engineer. publisher to be defined (2015)
2. Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)
3. Capozucca, A., Ries, B.: Excalibur - Design and Implementation of an Eclipse CASE tool for the Messir Method . publisher to be defined (2015)
4. Sommerville, I.: Software Engineering. 9 edn. Addison-Wesley, Harlow, England (2010)
5. Page-Jones, M.: Fundamentals of Object-oriented Design in UML. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)