



iCrash :
A Crisis Management Case Study
MESSIR Analysis Document
- v 1.4 -

(Report type: Simulation)

Friday 3rd March, 2017 - 13:07

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 **Messip** methodology [?]. 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 **Messip** 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 **Messip** methodology [?]. 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 ?. The **system operation** triggered by events sent by the external **actors** 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 **Messip** method, the information provided in this section is intended to present the system for which the **Messip** 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 **Messip** 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 **Messip** 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 an 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 is related to as a victim or witness.

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

- to provide as much details as possible 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 communicate with the *iCrash* system.

2.1.3 Coordinators

A coordinator is an employee of the ABC company being responsible of handling one or several crises. 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 an employee of the ABC company being responsible of administrating the *iCrash* system. The objectives of an administrator 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

Any system has a `Creator` stakeholder which is a technician who is installing the *iCrash* system on the targeted deployment infrastructure.

The objectives of a `Creator` are:

- to install the *iCrash* system
- to define the values for the initial system's state
- 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 [?] for more details).

Among all the stakeholders presented in the previous section, we can determine five types of direct actors¹:

- `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.

2.3 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the **Messir** method and inspired by the standard Cokburn template [?].

2.3.1 Use Cases

2.3.1.1 summary-suDeployAndRun

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 situations depending on alerts received.

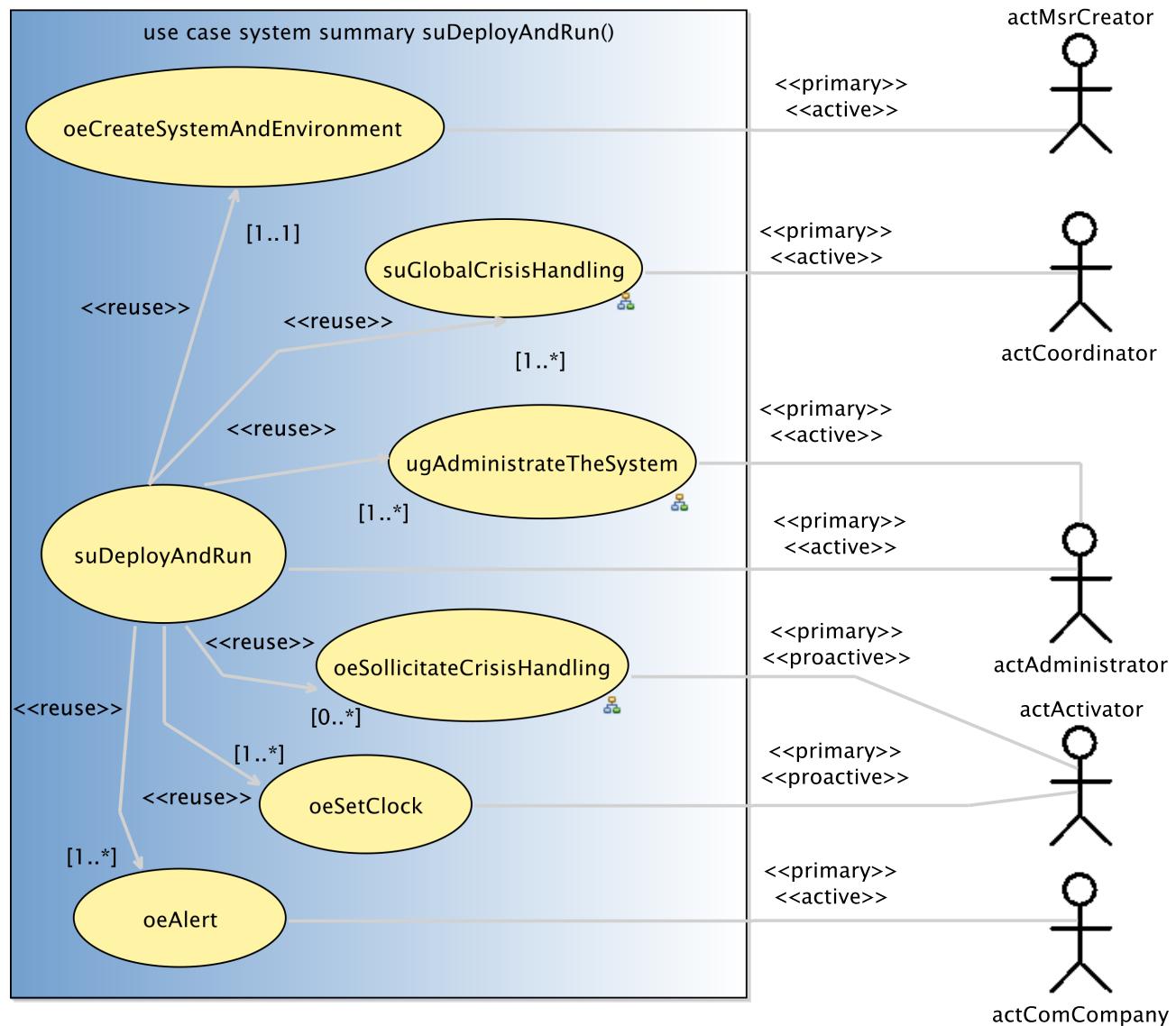
¹The naming conventions in **Messir** propose to start each type name by lowercase letters indicating the meta model type used (i.e. act for actors, ct for class type,). In addition to ease the reading it makes the translational semantics into Prolog code more straightforward.

| USE-CASE DESCRIPTION | |
|--|--|
| <i>Name</i> | suDeployAndRun |
| <i>Scope</i> | system |
| <i>Level</i> | summary |
| Primary actor(s) | |
| 1 | actAdministrator [active] |
| Secondary actor(s) | |
| 1 | actMsrCreator [active] |
| 2 | actCoordinator [active, multiple] |
| 3 | actActivator [proactive] |
| 4 | 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 situations depending on alerts received. | |
| Reuse | |
| 1 | <u>oeCreateSystemAndEnvironment [1..1]</u> |
| 2 | <u>ugAdministrateTheSystem [1..*]</u> |
| 3 | <u>suGlobalCrisisHandling [1..*]</u> |
| 4 | <u>oeSetClock [1..*]</u> |
| 5 | <u>oeSollicitateCrisisHandling [0..*]</u> |
| 6 | <u>oeAlert [1..*]</u> |
| Protocol condition(s) | |
| 1 | the iCrash system has never been deployed and used |
| Pre-condition(s) | |
| 1 | none |
| Main post-condition(s) | |
| 1 | the iCrash system has been created and has handled the crisis situations for which it received alerts through the communication company. |
| Main Steps | |
| a | the actor actMsrCreator executes the <u>oeCreateSystemAndEnvironment</u> use case |
| b | the actor actAdministrator executes the <u>ugAdministrateTheSystem</u> use case |
| c | the actor actComCompany executes the <u>oeAlert</u> use case |
| d | the actor actActivator executes the <u>oeSetClock</u> use case |
| e | the actor actActivator executes the <u>oeSollicitateCrisisHandling</u> use case |
| f | the actor actCoordinator executes the <u>suGlobalCrisisHandling</u> use case |
| Steps Ordering Constraints | |
| 1 | step (a) must be always the first step. |
| 2 | step (f) can be executed by different actCoordinator actors. |
| 3 | if (e) then previously (d). |

Figure 2.1 shows the use case diagram for the suDeployAndRun summary use case

2.3.1.2 summary-suGlobalCrisisHandling

the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis.

Figure 2.1: `suDeployAndRun` summary use case

| USE-CASE DESCRIPTION | |
|---|---|
| <i>Name</i> | suGlobalCrisisHandling |
| <i>Scope</i> | system |
| <i>Level</i> | summary |
| Primary actor(s) | |
| 1 | actCoordinator [active] |
| Goal(s) description | |
| the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis. | |
| Reuse | |
| 1 | ugSecurelyUseSystem [1..*] |
| 2 | ugMonitor [1..*] |
| 3 | ugManageCrisis [1..*] |
| Protocol condition(s) | |
| 1 | the iCrash system has been deployed |
| 2 | the coordinator actor 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 coordinator on existing alerts or crisis OR the coordinator requested an updated status on existing alerts or crisis. |
| Main Steps | |
| a | the actor actCoordinator executes the ugSecurelyUseSystem use case |
| b | the actor actCoordinator executes the ugMonitor use case |
| c | the actor actCoordinator executes the ugManageCrisis use case |
| Steps Ordering Constraints | |
| 1 | steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have their protocol constrained by steps of (a)). |
| 2 | steps (a) (b) and (c) can be executed multiple times. |

Figure 2.2 shows the use case diagram for the suGlobalCrisisHandling user goal use case

2.3.1.3 usergoal-ugAdministateTheSystem

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.

| USE-CASE DESCRIPTION | |
|--|---------------------------|
| <i>Name</i> | ugAdministateTheSystem |
| <i>Scope</i> | system |
| <i>Level</i> | usergoal |
| Primary actor(s) | |
| 1 | actAdministrator [active] |
| 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. | |

continues in next page ...

... Use-Case Description table continuation

| |
|--|
| Reuse |
| 1 <u>ugSecurelyUseSystem [1..*]</u> |
| 2 <u>oeAddCoordinator [1..*]</u> |
| 3 <u>oeDeleteCoordinator [0..*]</u> |
| Protocol condition(s) |
| 1 the iCrash system has been deployed |
| Pre-condition(s) |
| 1 none |
| Main post-condition(s) |
| 1 modifications have been made to the system and its environment concerning existing or new coordinators. |
| Main Steps |
| a the actor <code>actAdministrator</code> executes the <u>ugSecurelyUseSystem</u> use case |
| b the actor <code>actAdministrator</code> executes the <u>oeAddCoordinator</u> use case |
| c the actor <code>actAdministrator</code> executes the <u>oeDeleteCoordinator</u> use case |
| Steps Ordering Constraints |
| 1 steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have their protocol constrained by steps of (a)). |
| 2 steps (a) (b) and (c) can be executed multiple times. |

Figure 2.3 shows the use case diagram for the ugAdministrateTheSystem user goal use case

2.3.1.4 usergoal-ugManageCrisis

The goal is to do an action that makes the handling of a crisis or an alert progress.

| USE-CASE DESCRIPTION | |
|---|-------------------------------------|
| Name | ugManageCrisis |
| Scope | system |
| Level | usergoal |
| Primary actor(s) | |
| 1 | actCoordinator[active] |
| Goal(s) description | |
| The goal is to do an action that makes the handling of a crisis or an alert progress. | |
| Reuse | |
| 1 | <u>oeValidateAlert [0..*]</u> |
| 2 | <u>oeSetCrisisStatus [0..*]</u> |
| 3 | <u>oeSetCrisisHandler [0..*]</u> |
| 4 | <u>oeReportOnCrisis [0..*]</u> |
| 5 | <u>oeCloseCrisis [0..*]</u> |
| 6 | <u>oeInvalidateAlert [0..*]</u> |
| Protocol condition(s) | |
| 1 | the iCrash system has been deployed |
| Pre-condition(s) | |
| 1 | none |
| Main post-condition(s) | |

continues in next page ...

... Use-Case Description table continuation

| | |
|-----------------------------------|---|
| 1 | there exist one alert or one crisis whose related information has been changed. |
| Main Steps | |
| a | the actor actCoordinator executes the <u>oeValidateAlert</u> use case |
| b | the actor actCoordinator executes the <u>oeSetCrisisStatus</u> use case |
| c | the actor actCoordinator executes the <u>oeSetCrisisHandler</u> use case |
| d | the actor actCoordinator executes the <u>oeReportOnCrisis</u> use case |
| e | the actor actCoordinator executes the <u>oeCloseCrisis</u> use case |
| f | the actor actCoordinator executes the <u>oeInvalidateAlert</u> use case |
| Steps Ordering Constraints | |
| 1 | managing a crisis is doing one of the indicated use cases. |

Figure 2.4 shows the use case diagram for the ugManageCrisis user goal use case

2.3.1.5 usergoal-ugMonitor

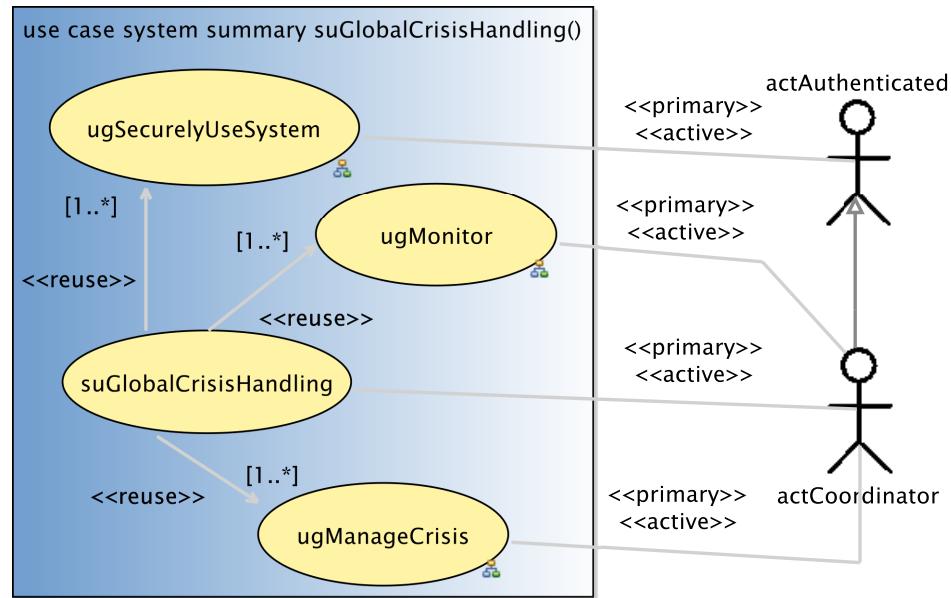
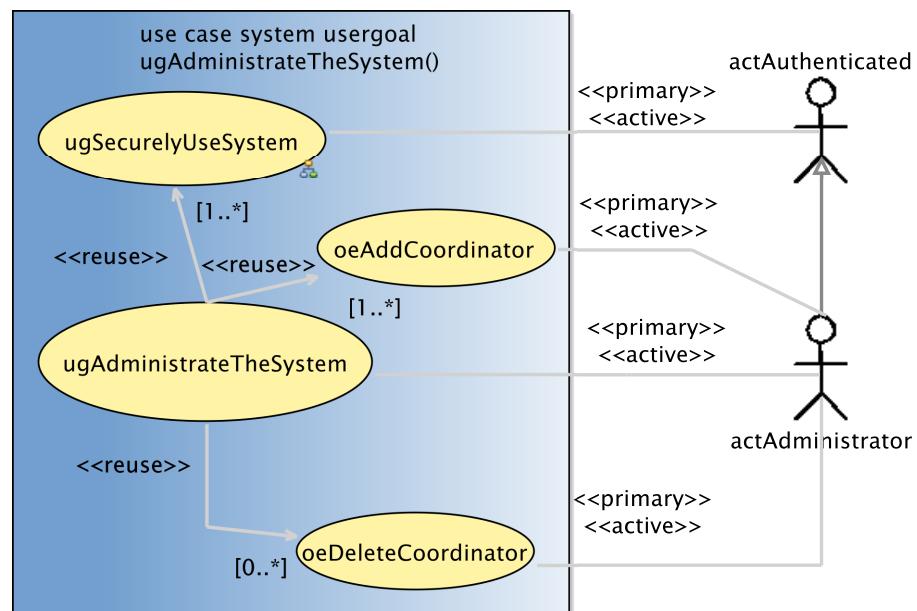
the actCoordinator's goal is to get the detailed list of existing crisis or alerts to decide on next actions to undertake.

| USE-CASE DESCRIPTION | |
|--|--|
| Name | ugMonitor |
| Scope | system |
| Level | usergoal |
| Primary actor(s) | |
| 1 | actCoordinator[active] |
| Goal(s) description | |
| the actCoordinator's goal is to get the detailed list of existing crisis or alerts to decide on next actions to undertake. | |
| Reuse | |
| 1 | <u>oeGetCrisisSet</u> [0..*] |
| 2 | <u>oeGetAlertsSet</u> [0..*] |
| Protocol condition(s) | |
| 1 | the iCrash system has been deployed |
| Pre-condition(s) | |
| 1 | none |
| Main post-condition(s) | |
| 1 | none |
| Main Steps | |
| a | the actor actCoordinator executes the <u>oeGetAlertsSet</u> use case |
| b | the actor actCoordinator executes the <u>oeGetCrisisSet</u> use case |

Figure 2.5 shows the use case diagram for the ugMonitor user goal use case

2.3.1.6 usergoal-ugSecurelyUseSystem

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.

Figure 2.2: `suGlobalCrisisHandling` user goal use caseFigure 2.3: `ugAdministateTheSystem` user goal use case

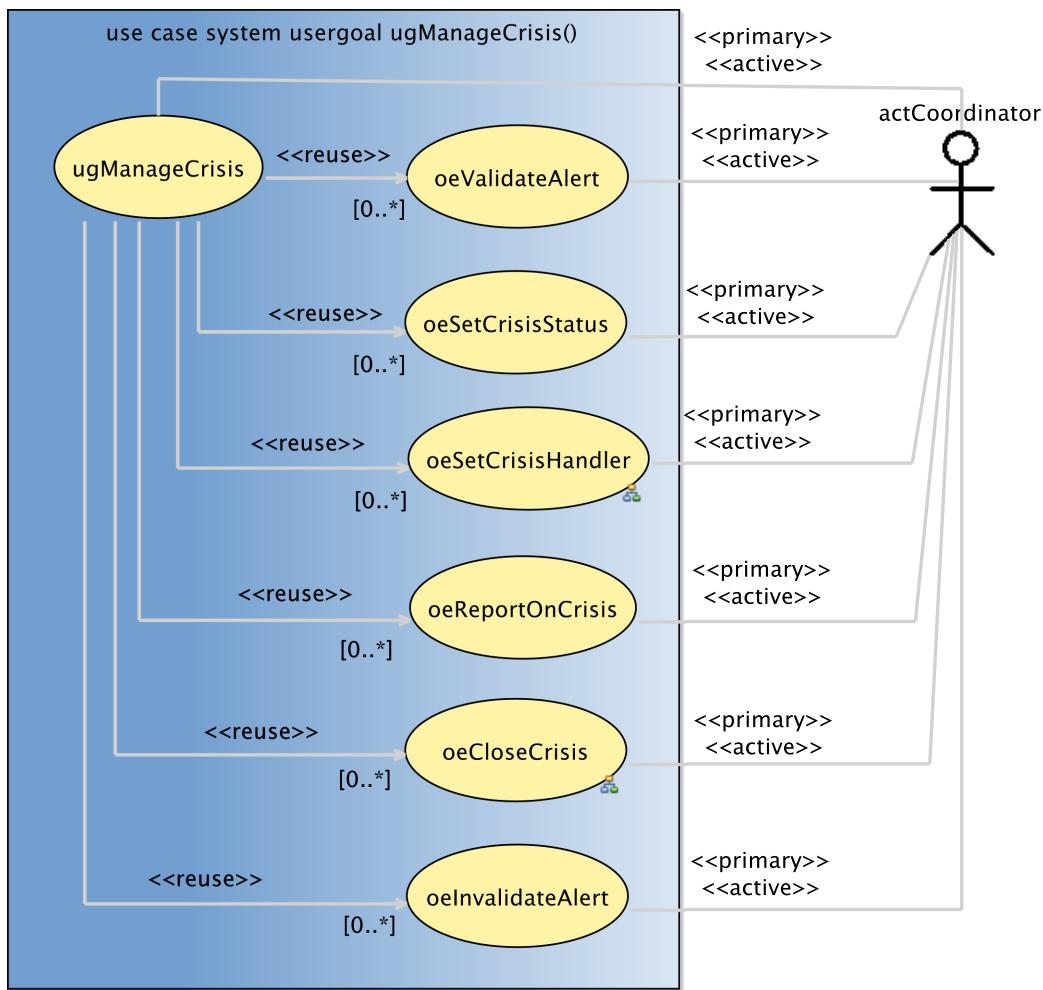


Figure 2.4: ugManageCrisis user goal use case

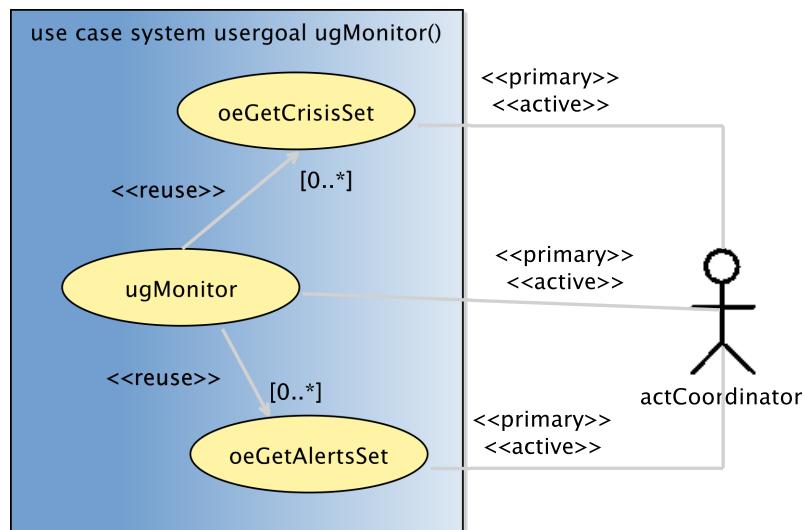


Figure 2.5: ugMonitor user goal use case

| USE-CASE DESCRIPTION | |
|-----------------------------------|--|
| Name | ugSecurelyUseSystem |
| Scope | system |
| Level | usergoal |
| <i>Primary actor(s)</i> | |
| 1 | actAuthenticated [active] |
| <i>Goal(s) description</i> | 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. |
| <i>Reuse</i> | |
| 1 | <u>oeLogin</u> [1..1] |
| 2 | <u>oeLogout</u> [1..1] |
| <i>Protocol condition(s)</i> | |
| 1 | the iCrash system has been deployed |
| <i>Pre-condition(s)</i> | |
| 1 | none |
| <i>Main post-condition(s)</i> | |
| 1 | the actAuthenticated is known by the system not to be logged. |
| <i>Main Steps</i> | |
| a | the actor actAuthenticated executes the <u>oeLogin</u> use case |
| b | the actor actAuthenticated executes the <u>oeLogout</u> use case |
| <i>Steps Ordering Constraints</i> | |
| 1 | step (a) must always precede step (b). |

Figure 2.6 shows the use case diagram for the ugSecurelyUseSystem user goal use case

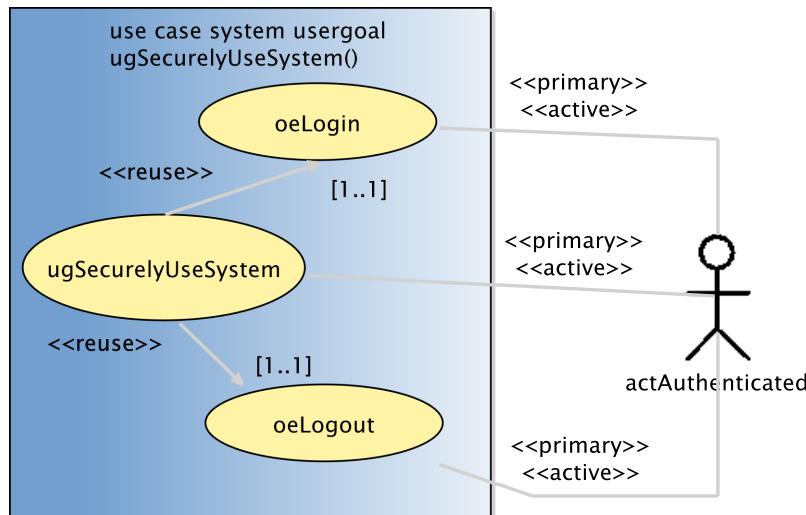


Figure 2.6: ugSecurelyUseSystem user goal use case

2.3.1.7 subfunction-oeSetCrisisHandler

goal is to declare himself as been the handler of a crisis having the specified id.

| USE-CASE DESCRIPTION | |
|---|-----------------------------------|
| Name | oeSetCrisisHandler |
| Scope | system |
| Level | subfunction |
| <i>Parameters</i> | |
| AdtCrisisID: | dtCrisisID 1 |
| <i>Primary actor(s)</i> | |
| 1 | actCoordinator [active] |
| <i>Secondary actor(s)</i> | |
| 1 | actCoordinator [passive] |
| 2 | actComCompany [passive, multiple] |
| <i>Goal(s) description</i> | |
| goal is to declare himself as been the handler of a crisis having the specified id. | |
| <i>Protocol condition(s)</i> | |
| 1 | |
| <i>Pre-condition(s)</i> | |
| 1 | |
| <i>Main post-condition(s)</i> | |
| 1 | |
| <i>Additional Information</i> | |
| none | |

Figure 2.7 shows the use case diagram for the oeSetCrisisHandler subfunction use case

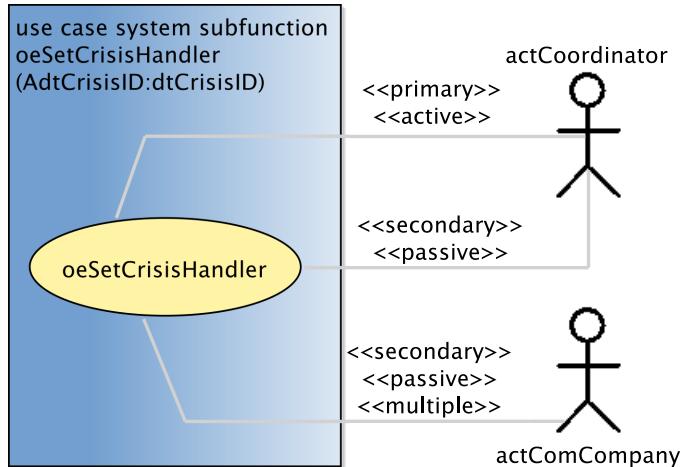


Figure 2.7: oeSetCrisisHandler subfunction use case

2.3.1.8 subfunction-oeSollicitateCrisisHandling

the actActivator's goal is to decrease the number of unhandled crisis.

| USE-CASE DESCRIPTION | |
|--|---|
| Name | oeSollicitateCrisisHandling |
| Scope | system |
| Level | subfunction |
| <i>Primary actor(s)</i> | |
| 1 | actActivator [proactive] |
| <i>Secondary actor(s)</i> | |
| 1 | actCoordinator [passive, multiple] |
| 2 | actAdministrator [passive] |
| <i>Goal(s) description</i> | |
| the actActivator's goal is to decrease the number of unhandled crisis. | |
| <i>Protocol condition(s)</i> | |
| 1 | the iCrash system has been deployed. |
| 2 | there exist some crisis still pending and for which no solicitation has been sent to the administrator and the coordinators for more than a predefined maximum delay. |
| <i>Pre-condition(s)</i> | |
| 1 | none |
| <i>Main post-condition(s)</i> | |
| 1 | a simple text message ieMessage('There are alerts not treated since more than the defined delay. Please REACT !') is sent to the system administrator and to all the coordinators of the environment for each crisis that is known to be not handled and for which no solicitation has been sent to the administrator and the coordinators for more than a predefined maximum delay.) |
| 2 | the reminder period for the concerned crisis is initialized. |

Figure 2.8 shows the use case diagram for the oeSollicitateCrisisHandling subfunction use case

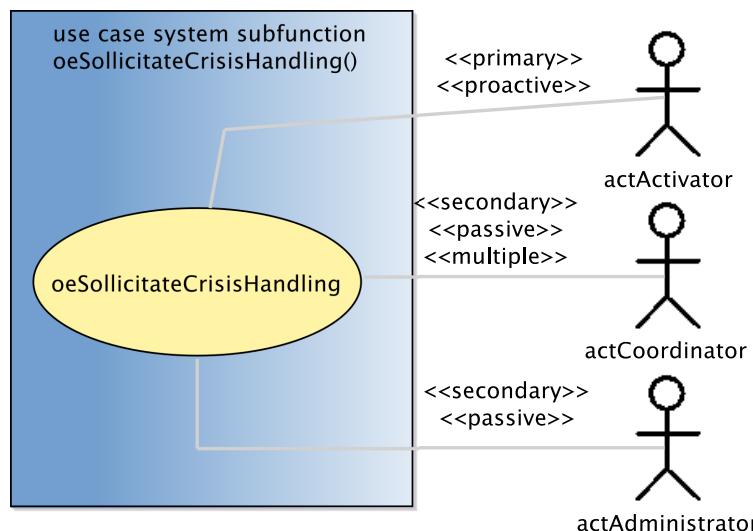


Figure 2.8: oeSollicitateCrisisHandling subfunction use case

2.3.2 Use Case Instance(s)

2.3.2.1 Use-Case Instance - uciSimpleAndCompletePart01:suDeployAndRun

First part of a use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

| SUMMARY USE-CASE INSTANCE | |
|---|--|
| <i>Instantiated Use Case</i> | |
| <code>suDeployAndRun</code> | |
| <i>Instance ID</i> | |
| <code>uciSimpleAndCompletePart01</code> | |
| <i>Remarks</i> | |
| a | shows the system initialization and the first administrative tasks by the administrator. |
| b | The unique and always existing <code>actMsrCreator</code> actor instance (named here <code>theCreator</code>) requests the initialization of the system and its environment (made of one administrator identified here by <code>bill</code>), one activator actor (identified by <code>theClock</code>) and indicating that the number of communication company actor instances for the system's environment is 4 (one of them is identified here by <code>tango</code>) |
| c | the administrator logs in to initialize a coordinator |
| d | an alert is received. Time is going on without having the coordinator handling the alert which let's the proactive actor trigger the automatic sollicitation of crisis handling. |
| e | this first part stops before the coordinator logs in the system. |

Figure 2.9 shows the sequence diagram representing the first part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

2.3.2.2 Use-Case Instance - uciSimpleAndCompletePart02:suDeployAndRun

Second part of a simple and complete use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

| SUMMARY USE-CASE INSTANCE | |
|---|--|
| <i>Instantiated Use Case</i> | |
| <code>suDeployAndRun</code> | |
| <i>Instance ID</i> | |
| <code>uciSimpleAndCompletePart02</code> | |
| <i>Remarks</i> | |
| a | starts when the coordinator logs in the system until the full handling of all the existing crisis. |
| b | shows an instantiated case of handling of a crisis by a coordinator until its closure after reporting. |

Figure 2.10 shows the sequence diagram representing the second part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

2.3.2.3 Use-Case Instance - uciugSecurelyUseSystem:ugSecurelyUseSystem

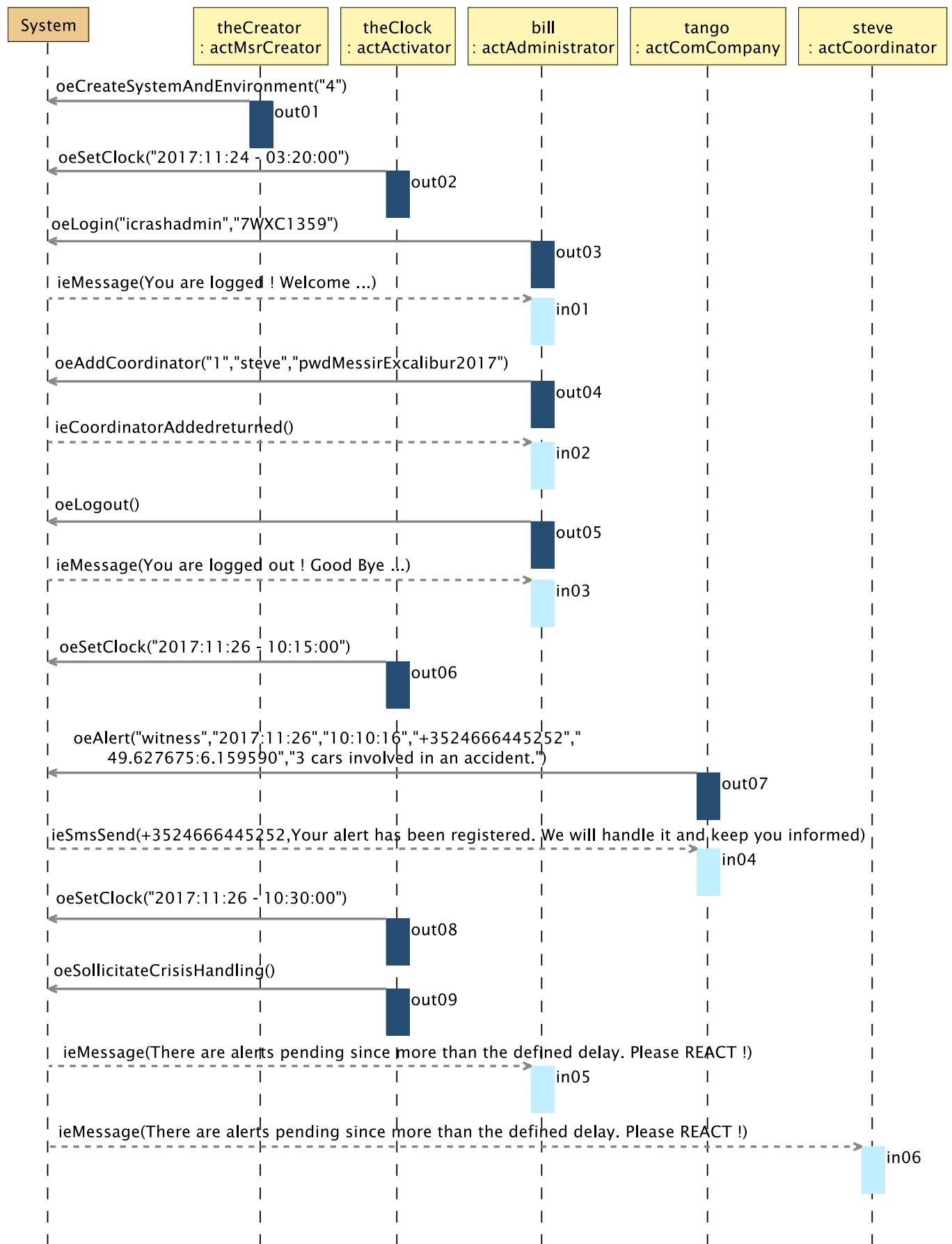


Figure 2.9: uci-suDeployAndRun-uciSimpleAndComplete-Part01

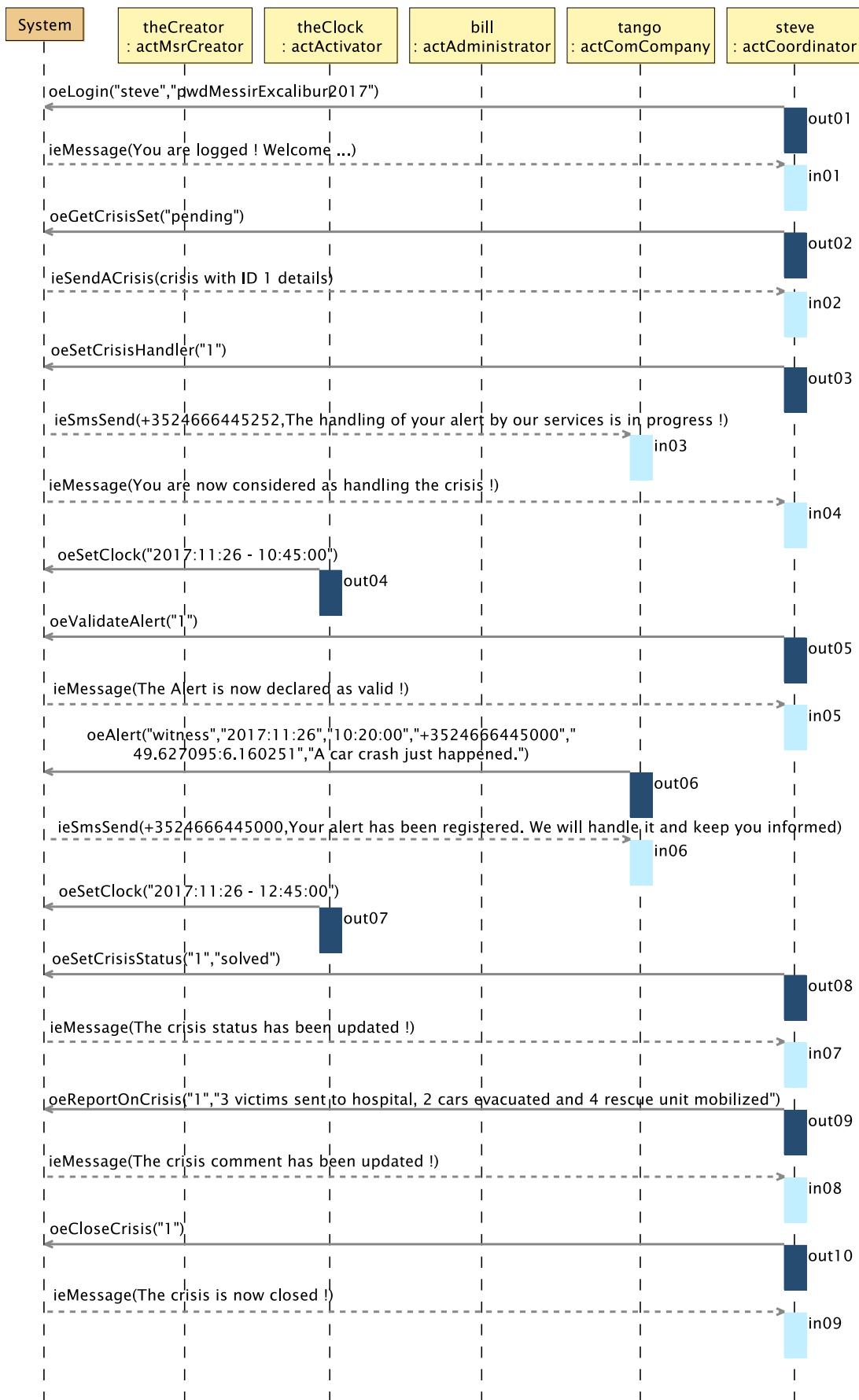


Figure 2.10: uci-suDeployAndRun-uciSimpleAndComplete-Part02 use case instance sequence diagram

| USERGOAL USE-CASE INSTANCE |
|---|
| <i>Instantiated Use Case</i> ugSecurelyUseSystem |
| <i>Instance ID</i> uciugSecurelyUseSystem |

Figure 2.11



Figure 2.11:

Chapter 3

Environment Model

We provide below the view(s) defined for the **Messip** environment model (cf. [?]) of the system.

3.1 Local view 01

Figure 3.1 shows the local view giving the second part 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 local view giving the second part 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 shows the local view for the administrator actor and interfaces

3.4 Local view 04

Figure 3.4 shows the local view for the coordinator actor and interfaces

3.5 Local view 05

Figure 3.5 shows the local view for the authenticated actor and interfaces

3.6 Global view 01

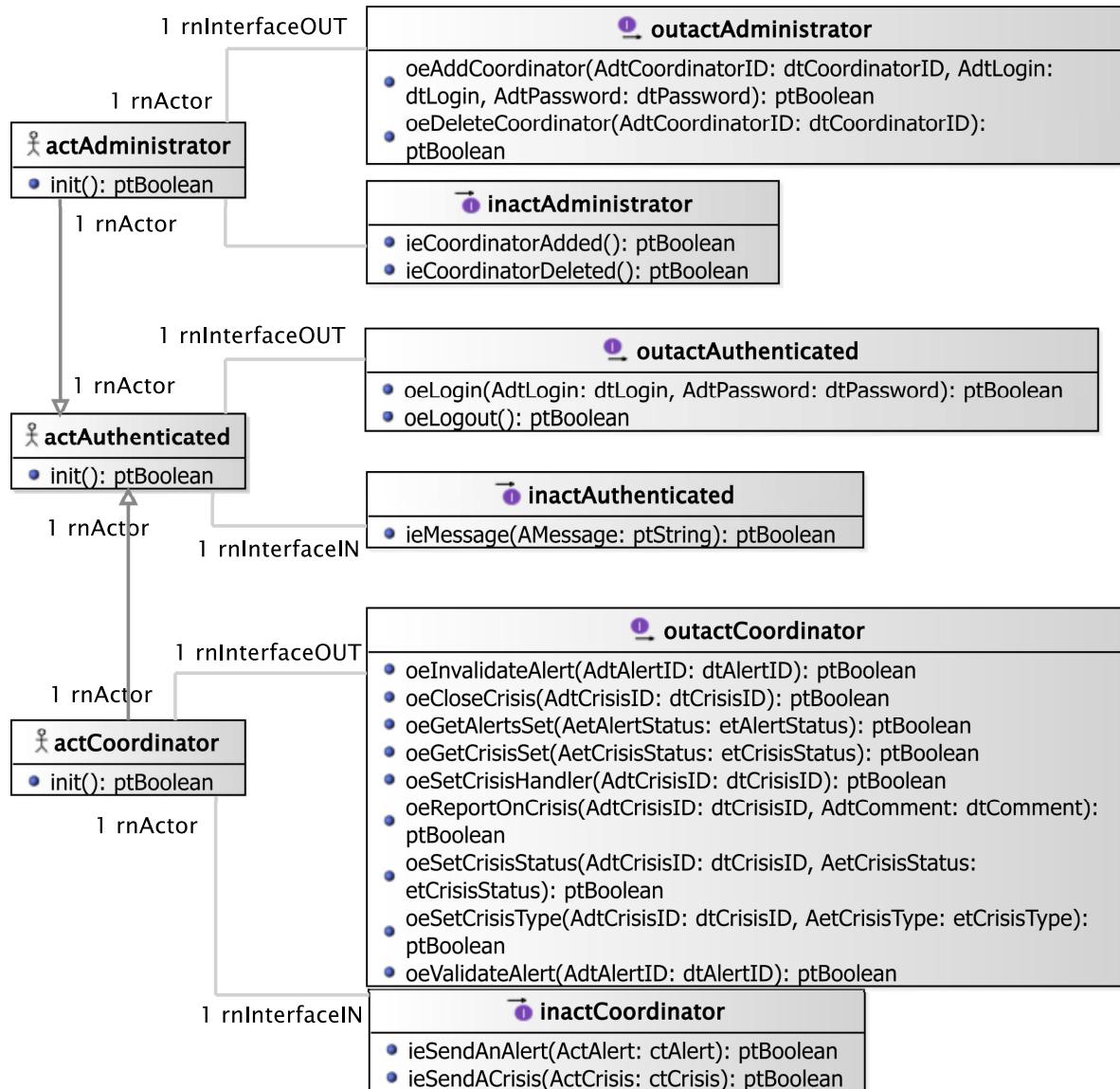


Figure 3.1: Environment Model - Local View 01. environment model local view - Part 1.

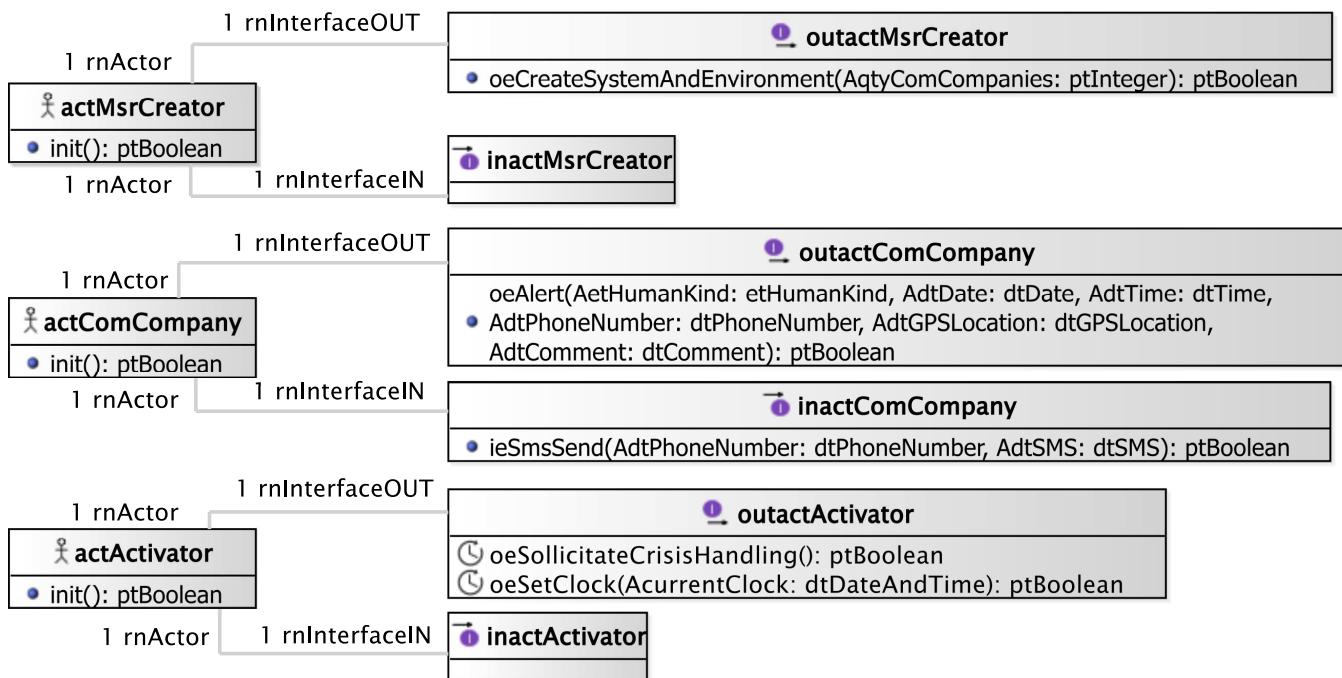


Figure 3.2: Environment Model - Local View 02. environment model local view - Part 2.

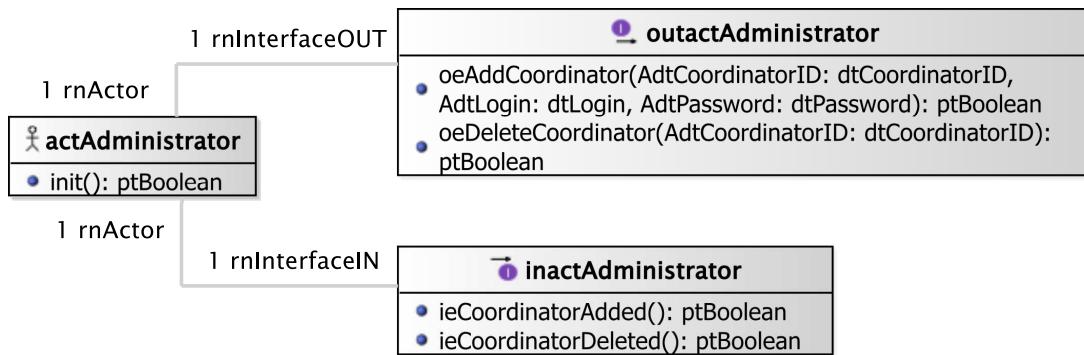


Figure 3.3: Environment Model - Local View 03. administrator actor environment model view.

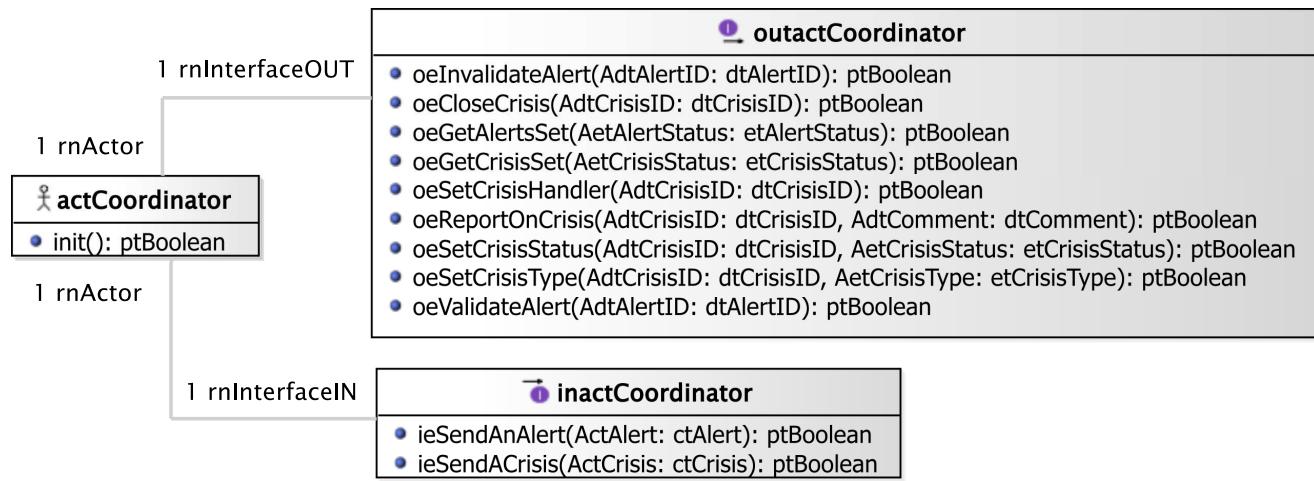


Figure 3.4: Environment Model - Local View 04. coordinator actor environment model view.

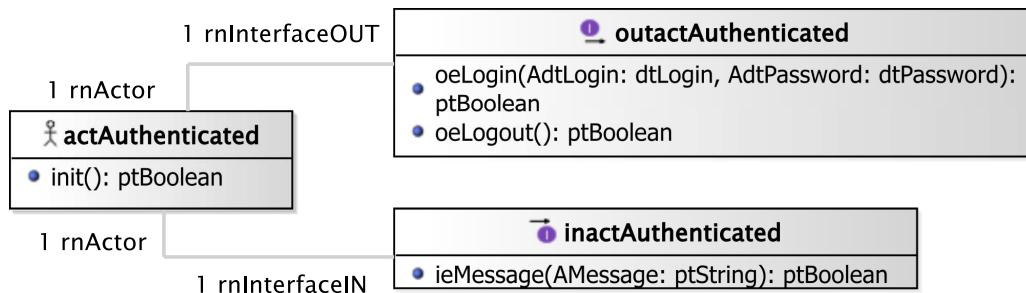


Figure 3.5: Environment Model - Local View 05. authenticated actor environment model local view.

Figure 3.6 shows a global view for all actors with their relationships with ctState

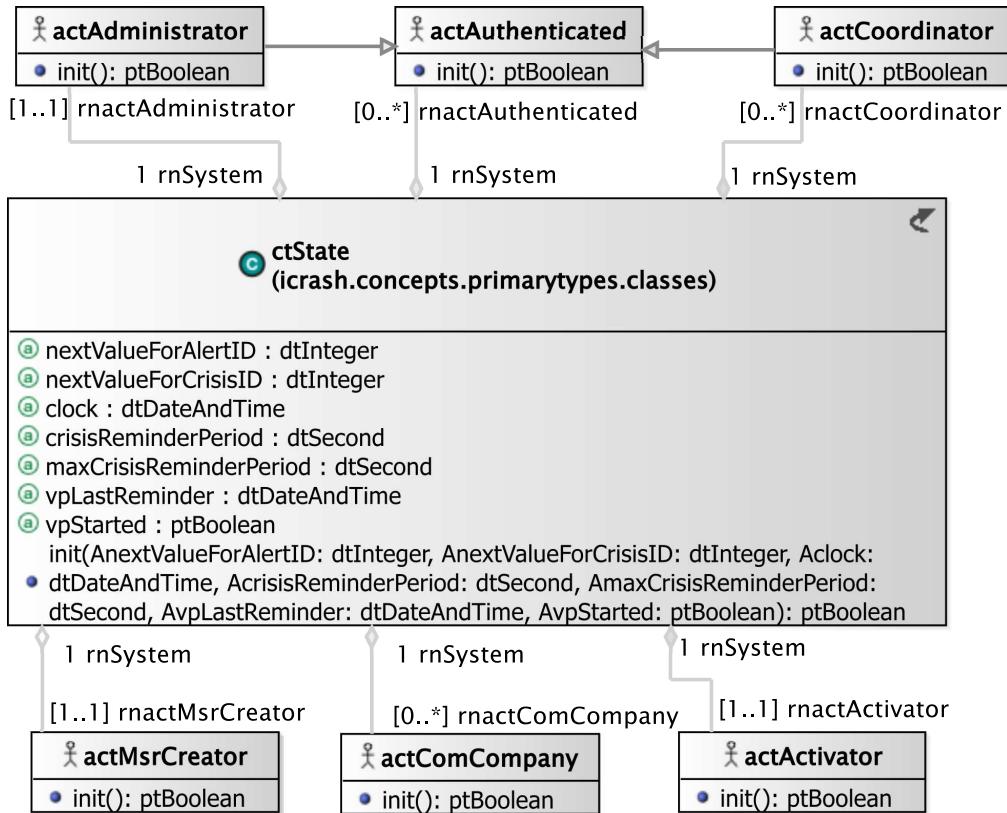


Figure 3.6: Environment Model - Global View 01. em-gv-01 environment model global view.

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 **actActivator** Actor

| ACTOR |
|---|
| actActivator |
| represents a logical actor for time automatic message sending based on system's or environment status. |
| OutputInterfaces |
| OUT 1 [proactive] oeSollicitateCrisisHandling () :ptBoolean used to avoid crisis to stay too long in an not handled status. OUT 2 [proactive] oeSetClock (AcurrentClock:dtDateAndTime) :ptBoolean used to update the system's time |

3.7.2 **actAdministrator** Actor

| ACTOR | |
|---|---|
| <i>actAdministrator</i> | |
| represents an actor responsible of administration tasks for the <i>iCrash</i> system. | |
| <i>Extends</i> | |
| icrash.environment.actAuthenticated | |
| <i>OutputInterfaces</i> | |
| OUT 1 | oeAddCoordinator (AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin, AdtPassword:dtPassword) :ptBoolean sent to add a new coordinator in the system's post state and environment's post state. |
| OUT 2 | oeDeleteCoordinator (AdtCoordinatorID:dtCoordinatorID) :ptBoolean sent to delete an existing coordinator in the system's post state and environment's post state. |
| <i>InputInterfaces</i> | |
| IN 1 | ieCoordinatorAdded () :ptBoolean its reception confirms the creation of the requested coordinator. |
| IN 2 | ieCoordinatorDeleted () :ptBoolean its reception confirms the deletion of the requested coordinator. |

3.7.3 **actAuthenticated** Actor

| ACTOR | |
|--|--|
| <i>actAuthenticated</i> | |
| abstract actor providing reusable input and output interfaces for actors that need to authenticate themselves. | |
| <i>OutputInterfaces</i> | |
| OUT 1 | oeLogin (AdtLogin:dtLogin, AdtPassword:dtPassword) :ptBoolean sent to request authorization to request access secured system operations. |
| OUT 2 | oeLogout () :ptBoolean sent to end the secured access to specific system operations. |
| <i>InputInterfaces</i> | |
| IN 1 | ieMessage (AMessage:ptString) :ptBoolean allows for receiving general textual messages. |

3.7.4 **actComCompany** Actor

| ACTOR | |
|---|--|
| <i>actComCompany</i> | |
| represents the communication company stakeholder ensuring the input/ouput of textual messages with humans having communication devices. | |
| <i>OutputInterfaces</i> | |
| OUT 1 | oeAlert (AetHumanKind:etHumanKind, AdtDate:dtDate, AdtTime:dtTime, AdtPhoneNumber:dtPhoneNumber, AdtGPSLocation:dtGPSLocation, AdtComment:dtComment) :ptBoolean sent to alert of a potential crisis situation. |
| <i>InputInterfaces</i> | |
| IN 1 | ieSmsSend (AdtPhoneNumber:dtPhoneNumber, AdtSMS:dtSMS) :ptBoolean |

continues in next page ...

...Actor table continuation

| |
|---|
| allows for receiving textual messages to be dispatched to the communication company customers having the provided phone number. |
|---|

3.7.5 actCoordinator Actor

| ACTOR | |
|--|--|
| <i>actCoordinator</i> | |
| represents actor responsible of handling one or several crisis for the <i>iCrash</i> system. | |
| <i>Extends</i> | |
| icrash.environment.actAuthenticated | |
| <i>OutputInterfaces</i> | |
| OUT 1 | oeInvalidateAlert (AdtAlertID:dtAlertID) :ptBoolean sent to indicate that an alert should be considered as closed. |
| OUT 2 | oeCloseCrisis (AdtCrisisID:dtCrisisID) :ptBoolean sent to indicate that a crisis should be considered as closed. |
| OUT 3 | oeGetAlertsSet (AetAlertStatus:etAlertStatus) :ptBoolean sent to request all the ctAlert instances having a specific status. |
| OUT 4 | oeGetCrisisSet (AetCrisisStatus:etCrisisStatus) :ptBoolean sent to request all the ctCrisis instances having a specific status. |
| OUT 5 | oeSetCrisisHandler (AdtCrisisID:dtCrisisID) :ptBoolean sent to declare himself as been the handler of a crisis having the specified id. |
| OUT 6 | oeReportOnCrisis (AdtCrisisID:dtCrisisID, AdtComment:dtComment) :ptBoolean sent to update the textual information available for a specific handled crisis. |
| OUT 7 | oeSetCrisisStatus (AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisStatus) :ptBoolean sent to define the handling status of a specific crisis. |
| OUT 8 | oeSetCrisisType (AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType) :ptBoolean sent to define the gravity type of a specific crisis. |
| OUT 9 | oeValidateAlert (AdtAlertID:dtAlertID) :ptBoolean sent to indicate that a specific alert is not a fake. |
| <i>InputInterfaces</i> | |
| IN 1 | ieSendAnAlert (ActAlert:ctAlert) :ptBoolean allows for receiving a requested ctAlert instance. |
| IN 2 | ieSendACrisis (ActCrisis:ctCrisis) :ptBoolean allows for receiving a requested ctCrisis instance. |

3.7.6 actMsrCreator Actor

| ACTOR | |
|---|---|
| <i>actMsrCreator</i> | |
| Represents the creator stakeholder in charge of state and environment initialization. | |
| <i>OutputInterfaces</i> | |
| OUT 1 | oeCreateSystemAndEnvironment (AqtyComCompanies:ptInteger) :ptBoolean sent to request the initialization of the system's class instances and the environment actors instances. |

Chapter 4

Concept Model

4.1 PrimaryTypes-Classes

4.1.1 Local view 01

Figure 4.1 shows the local view on all the primary types class types.

4.1.2 Local view 02

Figure 4.2 shows the local view of the ctState primary type class type.

4.1.3 Local view 03

Figure 4.3 shows the local view of the ctAlert primary type class type.

4.1.4 Local view 04

Figure 4.4 shows the local view of the ctCrisis primary type class type.

4.1.5 Global view 01

Figure 4.5 shows the global view on primary types class types showing the association(s) types with the actor classes of the environment model.

4.2 PrimaryTypes-Datatypes

4.2.1 Local view 06

Figure 4.6

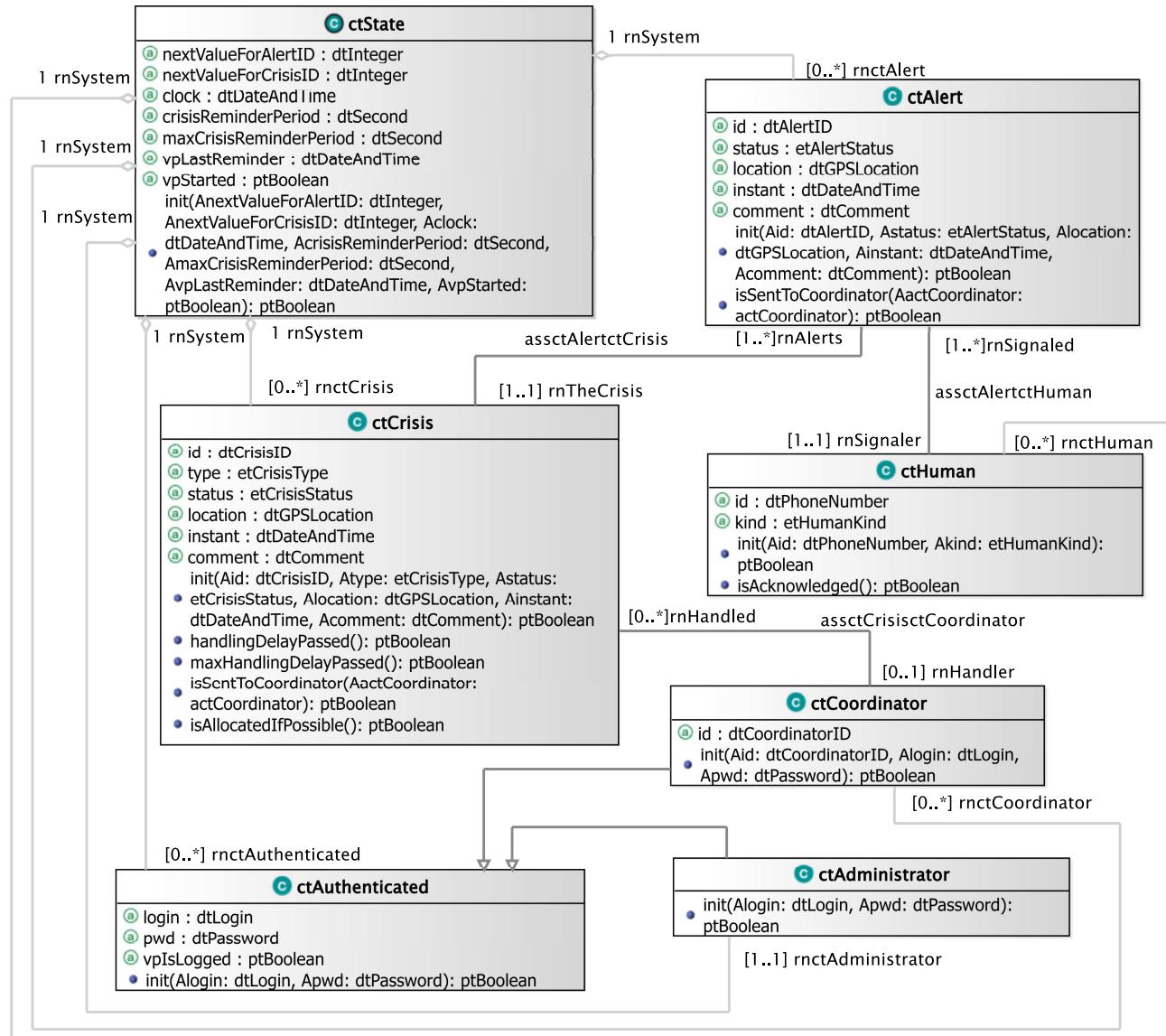


Figure 4.1: Concept Model - PrimaryTypes-Classes local view 01. Local view of all the primary types class types .

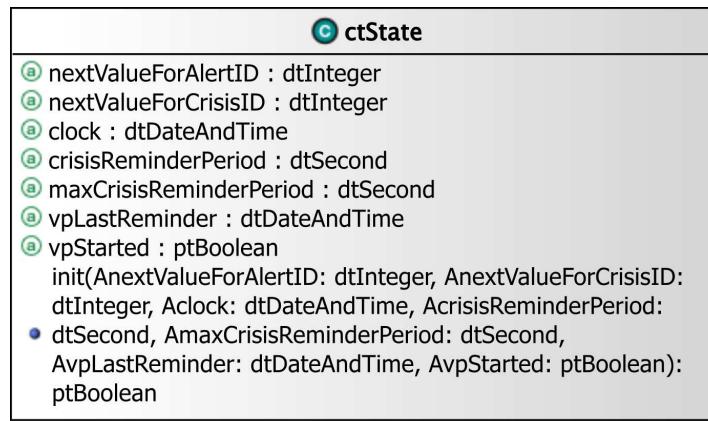


Figure 4.2: Concept Model - PrimaryTypes-Classes local view 02. local view of the ctState primary type.

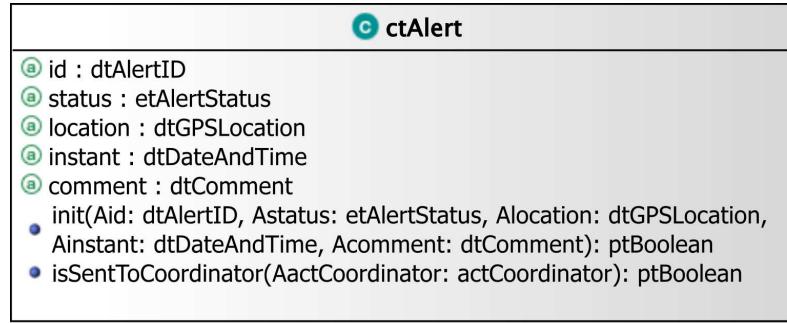


Figure 4.3: Concept Model - PrimaryTypes-Classes local view 03. local view of the ctAlert primary type.

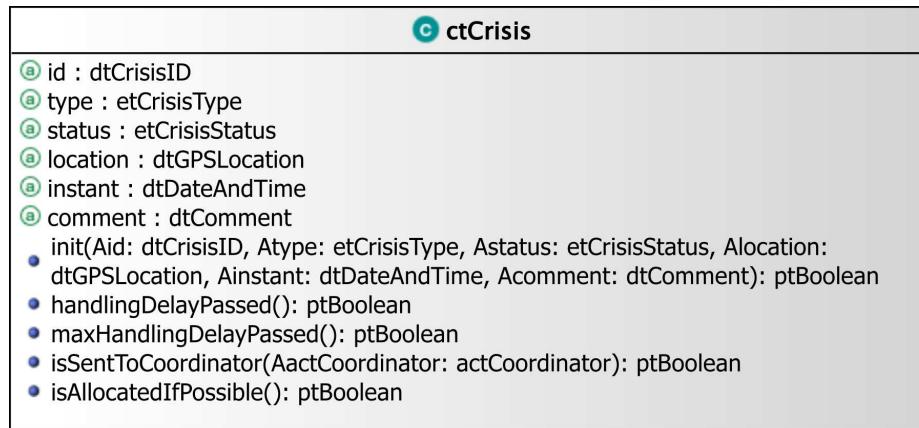


Figure 4.4: Concept Model - PrimaryTypes-Classes local view 04. local view of the ctCrisis primary type.

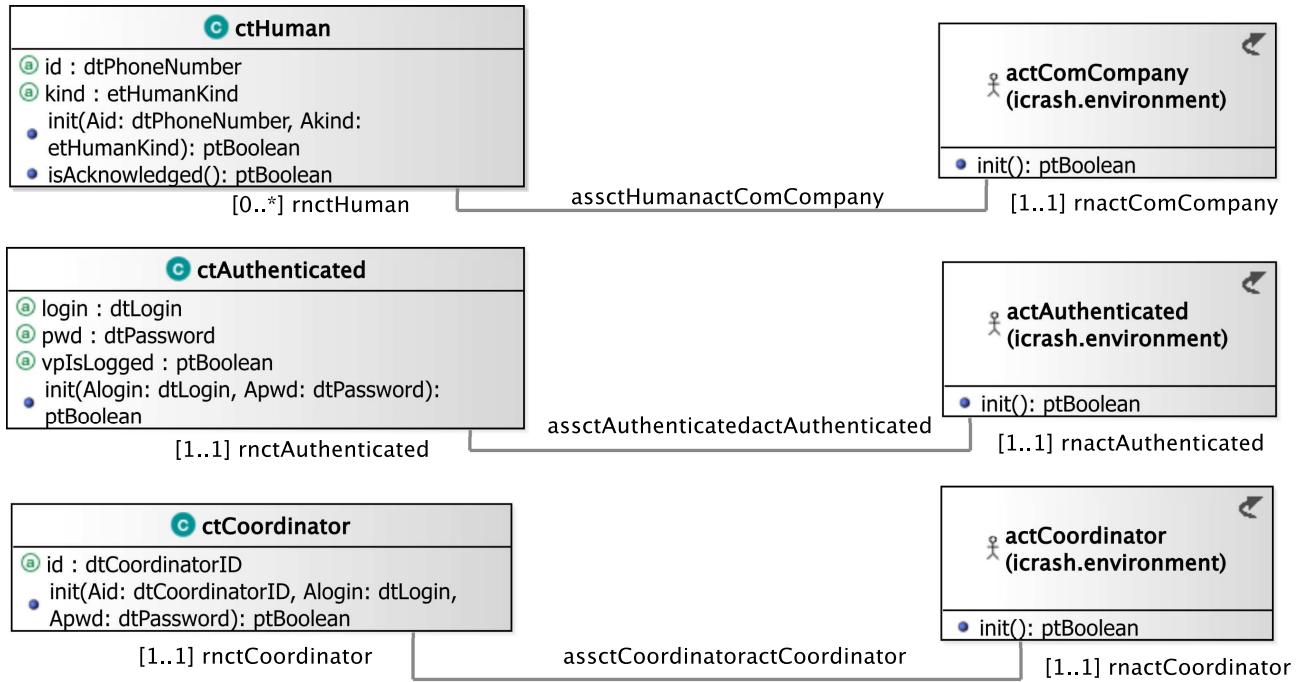


Figure 4.5: Concept Model - PrimaryTypes-Classes global view 01. Primary types class types global view - cm-pt-ct-gv-01 .

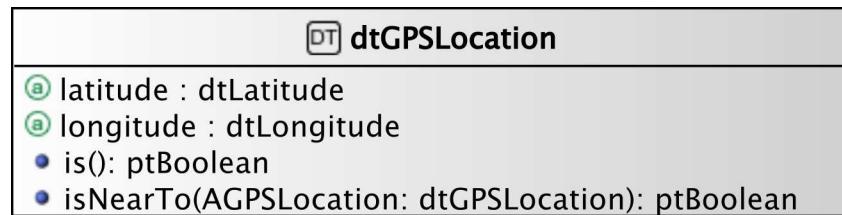


Figure 4.6: Concept Model - PrimaryTypes-Datatypes local view 06. .

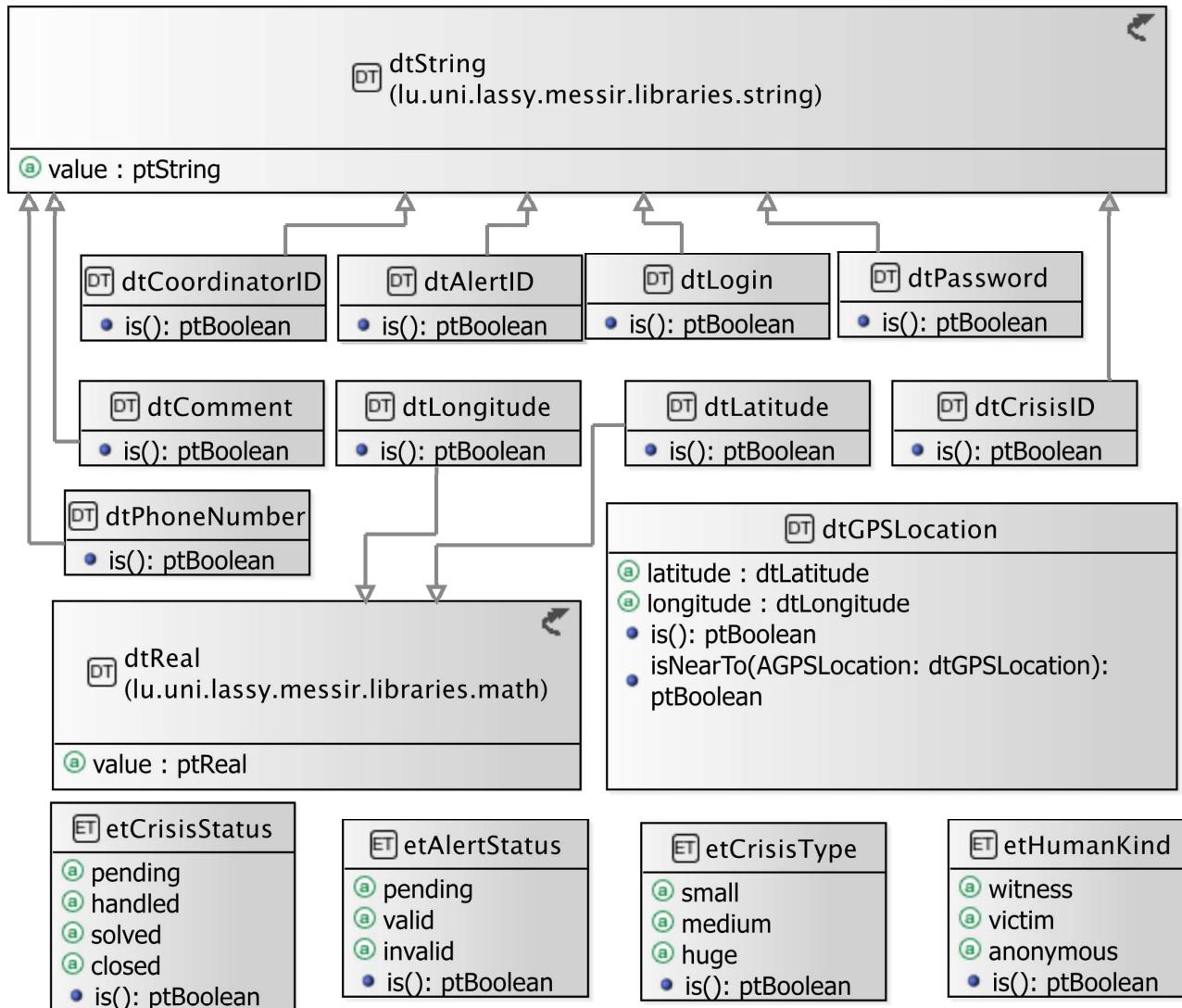


Figure 4.7: Concept Model - PrimaryTypes-Datatypes global view 01. global view of primary types datatype types - cm-pt-dt-gv-01 .

4.2.2 Global view 01

Figure 4.7 shows a global view on the *iCrash* primary types datatype types.

4.3 SecondaryTypes-Datatypes

4.3.1 Local view 01

Figure 4.8 shows the local view of the secondary types datatype types.

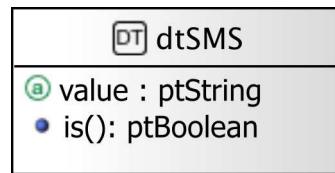


Figure 4.8: Concept Model - SecondaryTypes-Datatypes local view 01. Local view of the secondary types datatype types.

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 | |
|---|--|
| <i>ctAdministrator</i> | |
| used to characterize internally the entity that is responsible of administrating the <i>iCrash</i> system. | |
| <i>extends</i> | icrash.concepts.primarytypes.classes.ctAuthenticated |
| <i>operation</i> | init (Alogin:dtLogin, Apwd:dtPassword) :ptBoolean used to initialize the current object as a new instance of the ctAdministrator type. |
| <i>ctAlert</i> | |
| Used to model crisis alerts sent by any human having communication capability using communication companies belonging to the system's environment | |
| <i>attribute</i> | comment: dtComment a textual description providing unstructured information on the alert. |
| <i>attribute</i> | id: dtAlertID the alert unique identification information. |
| <i>attribute</i> | instant: dtDateAndTime the date and time at which the alert notification has been sent. |
| <i>attribute</i> | location: dtGPSLocation |

continues in next page ...

... Classes table continuation

| | |
|--|---|
| attribute | status: etAlertStatus the alert validation status |
| operation | init(Aid:dtAlertID, Astatus:etAlertStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment) :ptBoolean used to initialize the current object as a new instance of the ctAlert type. |
| operation | isSentToCoordinator(AactCoordinator:actCoordinator) :ptBoolean used to provide a given coordinator with current alert information. |
| ctAuthenticated | |
| used to model system's representation about actors that need to authenticate to access some specific functionalities. | |
| attribute | login: dtLogin an identifier for authentication. |
| attribute | pwd: dtPassword a key for authentication. |
| attribute | vpIsLogged: ptBoolean used to determine the access status. |
| operation | init(Alogin:dtLogin, Apwd:dtPassword) :ptBoolean used to initialize the current object as a new instance of the ctAuthenticated type. |
| ctCoordinator | |
| used to model system's representation about the actors that have the responsibility to handle alerts and crisis. | |
| extends | icrash.concepts.primarytypes.classes.ctAuthenticated |
| attribute | id: dtCoordinatorID a unique identification information. |
| operation | init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword) :ptBoolean used to initialize the current object as a new instance of the ctCoordinator type. |
| ctCrisis | |
| Used to model crisis that are inferred from the reception of at least one alert message. Crisis are entities that are handled by the <i>iCrash</i> system. | |
| attribute | comment: dtComment a textual description providing unstructured information on the crisis handling. |
| attribute | id: dtCrisisID the crisis unique identification information. |
| attribute | instant: dtDateAndTime the date and time at which the first related alert notification has been sent. |
| attribute | location: dtGPSLocation the position of the crisis equal to the one of the first alert received and associated to the crisis. |
| attribute | status: etCrisisStatus the crisis handling status. |
| attribute | type: etCrisisType an indication of the gravity of the crisis. |
| operation | handlingDelayPassed() :ptBoolean |

continues in next page ...

... Classes table continuation

| | |
|----------------|---|
| operation | used to determine if the crisis stood too longly in a pending status since last reminder. init (Aid:dtCrisisID, Atype:etCrisisType, Astatus:etCrisisStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment) :ptBoolean |
| operation | used to initialize the current object as a new instance of the ctAlert type. isAllocatedIfPossible () :ptBoolean |
| operation | used to allocate a crisis to a coordinator if any or to alert the administrator of crisis waiting to be handled. |
| operation | used to provide a given coordinator with current crisis information. isSentToCoordinator (AactCoordinator:actCoordinator) :ptBoolean |
| operation | used to determine if the crisis stood too longly in a pending status since its creation. maxHandlingDelayPassed () :ptBoolean |
| ctHuman | |
| | used to model system's representation about the indirect actors that has alerted of potential crisis. |
| attribute | id: dtPhoneNumber the number of the communication device used to send an alert to <i>iCrash</i> system. |
| attribute | kind: etHumanKind |
| | role with respect to the alert notified. |
| operation | init (Aid:dtPhoneNumber, Akind:etHumanKind) :ptBoolean init: used to initialize the current object as a new instance of the ctHuman type. |
| ctState | |
| | used to model the system. Each system specified using Messip must include a ctState class for which there is only one instance at any state of the abstract machine after creation. |
| attribute | clock: dtDateAndTime used to represent the system local time. |
| attribute | crisisReminderPeriod: dtSecond used to define the delay between two reminders after which a reminder must be sent to the administrator and to the known coordinators to encourage them to handle the crisis. |
| attribute | maxCrisisReminderPeriod: dtSecond used to define the maximum delay after which the crisis is randomly allocated to a coordinator if any or an alert message is sent to the administrator in order to encourage him to add coordinators. |
| attribute | nextValueForAlertID: dtInteger nextValueForAlertID: dtInteger: used to associate each alert declared with a unique identification value. |
| attribute | nextValueForCrisisID: dtInteger used to associate each crisis declared with a unique identification value. |
| attribute | vpLastReminder: dtDateAndTime date and time of the last reminder. |
| attribute | vpStarted: ptBoolean used to avoid reacting to an actor message if the system is not started (i.e. oeCreateSystemAndEnvironment not executed). |
| operation | init (AnextValueForAlertID:dtInteger, AnextValueForCrisisID:dtInteger, Aclock:dtDateAndTime, AcrisisReminderPeriod:dtSecond, AmaxCrisisReminderPeriod:dtSecond, AvpLastReminder:dtDateAndTime, AvpStarted:ptBoolean) :ptBoolean |

continues in next page ...

... Classes table continuation

| |
|--|
| used to initialize the current object as a new instance of the ctState type. |
|--|

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 | |
|---|---|
| <i>dtAlertID</i> | |
| extends | dtString |
| operation | is () :ptBoolean |
| | used to determine which strings are considered as valid alert identifiers. |
| <i>dtComment</i> | |
| a datatype made of a string value used to receive, store and send textual information about crisis and alerts. | |
| extends | dtString |
| operation | is () :ptBoolean |
| | used to determine which strings are considered as valid comments. |
| <i>dtCoordinatorID</i> | |
| A string used to identify coordinators. | |
| extends | dtString |
| operation | is () :ptBoolean |
| | used to determine which strings are considered as valid coordinators identifiers. |
| <i>dtCrisisID</i> | |
| A string used to identify crisis. | |
| extends | dtString |
| operation | is () :ptBoolean |
| | used to determine which strings are considered as valid crisis identifiers. |
| <i>dtGPSLocation</i> | |
| used to define coordinates of geographical positions on earth. It is defined a couple made of a latitude and a longitude. | |
| attribute | latitude: dtLatitude |
| | for the latitude part of the coordinate. |
| attribute | longitude: dtLongitude |
| | for the longitude part of the coordinate. |
| operation | is () :ptBoolean |
| | used to determine which couples are considered as valid dtGPSLocation values. |
| operation | isNearTo (AGPSLocation:dtGPSLocation) :ptBoolean |
| | used to determine if locations are considered enough close to be treated as equivalent in the application domain context. |
| <i>dtLatitude</i> | |
| used to define a latitude value of a geographical positions on earth. | |
| extends | dtReal |
| operation | is () :ptBoolean |
| | used to determine which strings are considered as valid dtLatitude. |
| <i>dtLogin</i> | |
| a login string used to authentify an <i>iCrash</i> user | |

continues in next page ...

... Datatypes table continuation

| | |
|---|-------------------------|
| <i>extends</i> | dtString |
| <i>operation</i> | is () :ptBoolean |
| used to determine which strings are considered as valid dtLogin. | |
| dtLongitude | |
| used to define a longitude value of a geographical positions on earth. | |
| <i>extends</i> | dtReal |
| <i>operation</i> | is () :ptBoolean |
| used to determine which strings are considered as valid dtLongitude. | |
| dtPassword | |
| a password string used to authentify an <i>iCrash</i> user | |
| <i>extends</i> | dtString |
| <i>operation</i> | is () :ptBoolean |
| used to determine which strings are considered as valid dtPassword. | |
| dtPhoneNumber | |
| a string used to store the phone number from the human declaring the crisis or the alert. | |
| <i>extends</i> | dtString |
| <i>operation</i> | is () :ptBoolean |
| used to determine which strings are considered as valid dtPhoneNumber. | |

| ENUMERATIONS |
|--|
| etAlertStatus |
| this type is used to indicate the different validation status of an alert. |
| <i>operation</i> |
| is () :ptBoolean |
| used to determine which litteral belongs to the enumeration. |
| etCrisisStatus |
| this type is used to indicate the different handling status of a crisis. |
| <i>operation</i> |
| is () :ptBoolean |
| used to determine which litteral belongs to the enumeration. |
| etCrisisType |
| this type is used to indicate the different types of a crisis. |
| <i>operation</i> |
| is () :ptBoolean |
| used to determine which litteral belongs to the enumeration. |
| etHumanKind |
| this type is used to indicate the kind of human that informs about a car crash crisis. |
| <i>operation</i> |
| is () :ptBoolean |
| used to determine which litteral belongs to the enumeration. |

4.4.3 Primary types - Association types descriptions

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

| UNDIRECTED ASSOCIATIONS |
|--|
| assctAlertctCrisis |
| a crisis is related to one or more alerts as the alerts judged to concern all the same crisis due to their location. An alert alerts exactly one crisis. |
| assctAlertctHuman |

continues in next page ...

... Undirected associations table continuation

alerts are notified by human through the communication company. We need to keep an internal representation of those human to allow for communication of alert handling.

assctAuthenticatedactAuthenticated

mainly used to determine if the login request of an authenticated actor can be granted based on the given credentials and the registered ones.

assctCoordinatoractCoordinator

frequent messages must be sent to coordinator especially in relation to crisis they handle.

assctCrisisctCoordinator

at any point in time we need to know if a coordinator is handling existing crisis or not.

assctHumanactComCompany

in order to communicate with humans who informed about potential crisis, we need to record the communication company to use to send them messages.

4.4.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

4.4.4.1 Primary types - Composition types descriptions

There are no composition types for the primary types.

4.4.5 Secondary types - Class types descriptions

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

4.4.6 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 | |
|---|---|
| <i>dtSMS</i> | |
| a datatype made of a string value used to send textual information to human mobile devices. | |
| attribute | <i>value: ptString</i> |
| | the textual information. |
| operation | <i>is():ptBoolean</i> |
| | used to determine which strings are considered as valid comments. |

4.4.7 Secondary types - Association types descriptions

There are no association types for the secondary types.

4.4.8 Secondary types - Aggregation types descriptions

There are no aggregation types for the secondary types.

4.4.9 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, in a primary or secondary type (class, datatype or enumeration types). The **Messip** OCL code listing is joined to the comment table.

5.1 Environment - Out Interface Operation Scheme for actActivator

5.1.1 Operation Model for oeSetClock

The oeSetClock operation has the following properties:

| OPERATION | |
|---|--|
| <i>oeSetClock[proactive]</i> | |
| An active message used to statically set the date and time information in the system's state. | |
| Parameters | |
| 1 | AcurrentClock: dtDateAndTime the date and time to be considered as the actual one. |
| Return type | |
| ptBoolean | |
| Pre-Condition (protocol) | |
| PreP 1 | the system is supposed to be created and initialized and the provided date and time value is greater than the one known by the system. |
| Pre-Condition (functional) | |
| PreF 1 | none |
| Post-Condition (functional) | |
| PostF 1 | the ctState instance post-state is updated to have its clock attribute equal to the given date and time. |
| Post-Condition (protocol) | |
| PostP 1 | none |

The listing 5.1 provides the **Messip** (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem: ctState in
```