University of Luxembourg
Dos Santos Loureiro Mike
Ribeiro Patrick
Texeira Mike

iCrash : Authentication- AccessRights- PointofInterest-variant
Messir Analysis Document
- v 0.2 -

(*Report type: Specification*)

Tuesday 29th May, 2018 - 12:33

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Overview

1.2 Purpose and recipients of the document

1.3 Application Domain

1.4 Definitions, acronyms and abbreviations

1.5 Document structure

Chapter 2

General Description

2.1 Domain Stakeholders

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).

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].

2.3.1 Use Cases

2.3.1.1 summary-suAddNewPI

The goal is the check the requested point of interest from the actor Person, treat the request and add the new PI to the system.

Use-Case Description	
Name	suAddNewPI
Scope	system
Level	summary
Primary actor(s)	
1	`actPerson[active]`
Secondary actor(s)	
1	`actMsrCreator[active]`
2	`actCoordinator[proactive]`
3	`actAdministrator[proactive]`
Goal(s) description	
The goal is the check the requested point of interest from the actor Person, treat the request and add the new PI to the system.	
Reuse	
1	`oeCreateSystemAndEnvironment [1..1]`
2	`ugSecurelyUseSystem [1..1]`
3	`oeSearchPI [1..*]`
4	`oeSendNewRequest [1..*]`
5	`ugCheckRequest [1..*]`
6	`ugManageRequest [1..*]`
7	`oeAddPI [1..*]`
Protocol condition(s)	
1	the iCrash system has never been deployed the coordinator actor involved in the use case has been declared by the actor actAdministrator
Pre-condition(s)	
1	none

continues in next page . . .

. . . Use-Case Description table continuation

Main post-condition(s)	
1	modifications have been made by the coordinator on existing requests (ignored attribute) the administrator has modified the status of the requests message solved request has been sent to the administrator the administrator added the new PI message added PI has been sent to the administrator and the person
Main Steps	
a	the actor `actMsrCreator` executes the `oeCreateSystemAndEnvironment` use case
b	the actor `actPerson` executes the `ugSecurelyUseSystem` use case
c	the actor `actPerson` executes the `oeSearchPI` use case
d	the actor `actPerson` executes the `oeSendNewRequest` use case
e	the actor `actCoordinator` executes the `ugSecurelyUseSystem` use case
f	the actor `actCoordinator` executes the `ugCheckRequest` use case
g	the actor `actCoordinator` executes the `ugSecurelyUseSystem` use case
h	the actor `actAdministrator` executes the `ugManageRequest` use case
i	the actor `actAdministrator` executes the `oeAddPI` use case
Steps Ordering Constraints	
1	step (a) is the first step to perform.
2	step (b), (d) and (f) must always be performed before (c), (e) and (g).
3	Subsequently, all the steps follow after step (a) is performed.
Additional Information	
none	

2.3.1.2 summary-suGenerateNewAlert

The goal is that the person searches for a point of interest, retrieves the GPS location and sends an Sms to the communication company with the needed information.

USE-CASE DESCRIPTION	
Name	suGenerateNewAlert
Scope	system
Level	summary
Primary actor(s)	
1	`actPerson[active, multiple]`
2	`actComCompany[active, multiple]`
Secondary actor(s)	
1	`actMsrCreator[active]`
Goal(s) description	
The goal is that the person searches for a point of interest, retrieves the GPS location and sends an Sms to the communication company with the needed information.	
Reuse	
1	`oeCreateSystemAndEnvironment [1..1]`
2	`ugSecurelyUseSystem [1..1]`
3	`oeSearchPI [1..*]`
4	`oeGetGPSLocation [1..*]`
5	`oeAlert [1..*]`
Protocol condition(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1	the iCrash system has never been deployed the point of interest already exists
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the actor person gets a match after searching for a PI the person gets a message with the GPS location of the PI a message that the Sms has been sent, is sent to the person the communication company gets a message with the detailed information of a potential crisis
Main Steps	
a	the actor `actMsrCreator` executes the `oeCreateSystemAndEnvironment` use case
b	the actor `actPerson` executes the `ugSecurelyUseSystem` use case
c	the actor `actPerson` executes the `oeSearchPI` use case
d	the actor `actPerson` executes the `oeGetGPSLocation` use case
e	the actor `actComCompany` executes the `oeAlert` use case
Steps Ordering Constraints	
1	step (a) and (b) must always be performed before all the other steps.
2	step (a) is the first step to perform.
3	Subsequently, all the steps follow after step (a) is performed.
Additional Information	
none	

2.3.1.3 summary-suGlobalCrisisHandling

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

Use-Case Description	
Name	suGlobalCrisisHandling
Scope	system
Level	summary
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
the actCoordinators goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis.	
Reuse	
1	`ugSecurelyUseSystem`
2	`ugManageCrisis`
3	`ugMonitor`
Protocol condition(s)	
1	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	
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.

continues in next page . . .

. . . Use-Case Description table continuation

Main Steps	
a	the actor actCoordinator executes the ugSecurelyUseSystem use case
b	the actor actCoordinator executes the ugManageCrisis use case
c	the actor actCoordinator executes the ugMonitor use case
Steps Ordering Constraints	
1	Step a must be executed before executing all other steps
Additional Information	
none	

2.3.1.4 summary-suScenarioPresentation

scenario shown in a presentation

USE-CASE DESCRIPTION	
Name	suScenarioPresentation
Scope	system
Level	summary
Primary actor(s)	
1	actMsrCreator[active]
2	actAdministrator[active]
3	actCoordinator[active]
4	actPerson[active]
5	actComCompany[active]
Goal(s) description	
scenario shown in a presentation	
Reuse	
1	oeCreateSystemAndEnvironment [1..1]
2	oeLoginWithCaptcha [1..*]
3	oeSearchPI [1..*]
4	oeSendNewRequest [1..*]
5	ugCheckRequest [1..*]
6	ugManageRequest [1..*]
7	oeSearchPI [1..*]
8	oeGetGPSLocation [1..*]
9	oeAlert [1..*]
10	ugAdministrateTheSystem [1..*]
11	ugMonitor [1..*]
12	ugManageCrisis [1..*]
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Main Steps	
a	the actor actMsrCreator executes the oeCreateSystemAndEnvironment use case
b	the actor actAdministrator executes the ugAdministrateTheSystem use case

continues in next page . . .

... Use-Case Description table continuation

c	the actor actCoordinator executes the oeLoginWithCaptcha use case
d	the actor actPerson executes the oeSearchPI use case
e	the actor actPerson executes the oeSendNewRequest use case
f	the actor actCoordinator executes the ugCheckRequest use case
g	the actor actAdministrator executes the ugManageRequest use case
h	the actor actPerson executes the oeSearchPI use case
i	the actor actPerson executes the oeGetGPSLocation use case
j	the actor actComCompany executes the oeAlert use case
k	the actor actCoordinator executes the ugMonitor use case
l	the actor actCoordinator executes the ugManageCrisis use case
Steps Ordering Constraints	
1	Step a must be executed before executing all other steps
2	Step b must follow step a
Additional Information	
none	

2.3.1.5 usergoal-ugAdministrateTheSystem

the actAdministrators 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	
Name	ugAdministrateTheSystem
Scope	system
Level	usergoal
Primary actor(s)	
1	actAdministrator[active]
Goal(s) description	
the actAdministrators 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..1]
2	oeAddCoordinator [1..*]
3	oeDeleteCoordinator [1..*]
4	oeUpdateCoordinatorAccessRights [1..*]
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	
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 actAdministrator executes the ugSecurelyUseSystem use case
b	the actor actAdministrator executes the oeAddCoordinator use case
c	the actor actAdministrator executes the oeDeleteCoordinator use case

continues in next page . . .

... Use-Case Description table continuation

d	the actor `actAdministrator` executes the `oeUpdateCoordinatorAccessRights` use case
Steps Ordering Constraints	
1	Step a must be executed before all other steps
2	Step b must be executed before executing steps c or d
Additional Information	
none	

2.3.1.6 usergoal-ugCheckRequest

The goal is to compare the new potential point of interest to the existing PIs in the system.

USE-CASE DESCRIPTION	
Name	ugCheckRequest
Scope	system
Level	usergoal
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
The goal is to compare the new potential point of interest to the existing PIs in the system.	
Reuse	
1	`ugSecurelyUseSystem [1..1]`
2	`oeGetAllRequests [1..1]`
3	`oeCheckAvailability [1..*]`
4	`oeDeliverRequest [1..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	there exist one request whose ignored information has been changed.
Main Steps	
a	the actor `actCoordinator` executes the `ugSecurelyUseSystem` use case
b	the actor `actCoordinator` executes the `oeGetAllRequests` use case
c	the actor `actCoordinator` executes the `oeCheckAvailability` use case
d	the actor `actCoordinator` executes the `oeDeliverRequest` use case
Steps Ordering Constraints	
1	step (a) must always be performed before all the other steps.
2	Subsequently, all the steps follow after step (a) is performed.
Additional Information	
none	

Figure 2.1 The coordinator checks if the requested point of interest is not already in the system.

2.3.1.7 usergoal-ugLoginWithCaptcha

The goal is to authenticate an actor as not being a machine/robot.

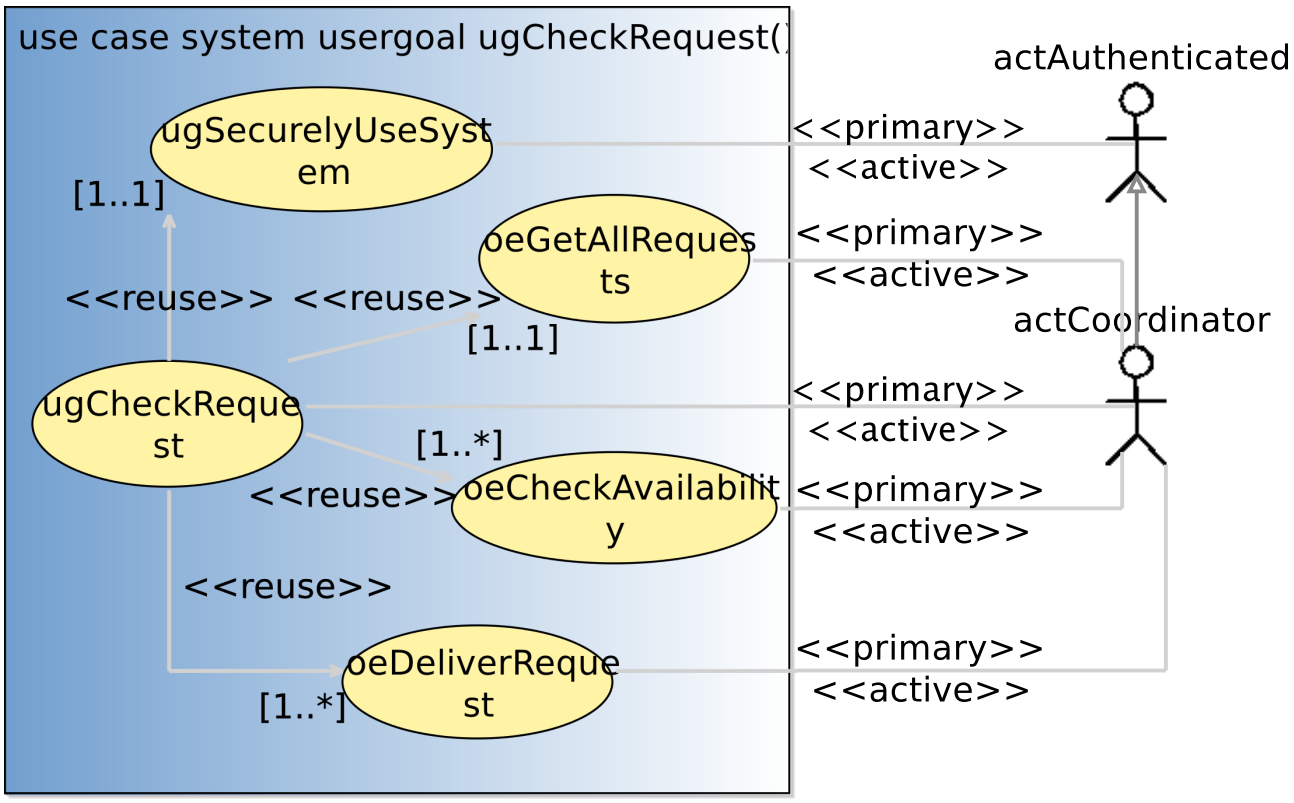

Figure 2.1: ugCheckRequest user goal use case

Use-Case Description	
Name	ugLoginWithCaptcha
Scope	system
Level	usergoal
Primary actor(s)	
1	actAuthenticated[active]
Goal(s) description	
The goal is to authenticate an actor as not being a machine/robot.	
Reuse	
1	oeLogin [3..*]
2	oeLoginWithCaptcha [1..*]
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Main Steps	
a	the actor actAuthenticated executes the oeLogin use case
b	the actor actAuthenticated executes the oeLogin use case
c	the actor actAuthenticated executes the oeLogin use case
d	the actor actAuthenticated executes the oeLoginWithCaptcha use case
Steps Ordering Constraints	
1	Step a to c must be executed before executing all other steps
2	Step d must be the last step
Additional Information	
none	

2.3.1.8 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	ugSecurelyUseSystem [1..1]
2	oeValidateAlert [1..*]
3	oeInvalidateAlert [1..*]
4	oeSetCrisisStatus [1..*]
5	oeSetCrisisType [1..*]
6	oeSetCrisisHandler [1..*]

continues in next page . . .

... Use-Case Description table continuation

7	oeReportOnCrisis [1..*]
8	oeCloseCrisis [1..*]
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Main Steps	
a	the actor actCoordinator executes the ugSecurelyUseSystem use case
b	the actor actCoordinator executes the oeValidateAlert use case
c	the actor actCoordinator executes the oeInvalidateAlert use case
d	the actor actCoordinator executes the oeSetCrisisHandler use case
e	the actor actCoordinator executes the oeSetCrisisStatus use case
f	the actor actCoordinator executes the oeSetCrisisType use case
g	the actor actCoordinator executes the oeReportOnCrisis use case
h	the actor actCoordinator executes the oeCloseCrisis use case
Steps Ordering Constraints	
1	Step a must be executed before executing all other steps
2	Step b must be executed before executing step d
3	Step d must executed before executing steps e, f, g and h
Additional Information	
none	

Figure 2.2 shows how crisis are managed.

2.3.1.9 usergoal-ugManageRequest

The goal is to do an action that makes the handling of a request progress.

USE-CASE DESCRIPTION	
Name	ugManageRequest
Scope	system
Level	usergoal
Primary actor(s)	
1	actAdministrator[active]
Goal(s) description	
The goal is to do an action that makes the handling of a request progress.	
Reuse	
1	ugSecurelyUseSystem [1..1]
2	oeGetAllRequestsFromCoordinator [1..1]
3	oeTreatRequest [1..*]
4	oeSolveRequest [1..*]
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	

continues in next page ...

. . . Use-Case Description table continuation

1	none
Main post-condition(s)	
1	there exist one request whose related information has been changed.
Main Steps	
a	the actor `actAdministrator` executes the `ugSecurelyUseSystem` use case
b	the actor `actAdministrator` executes the `oeGetAllRequestsFromCoordinator` use case
c	the actor `actAdministrator` executes the `oeTreatRequest` use case
d	the actor `actAdministrator` executes the `oeSolveRequest` use case
Steps Ordering Constraints	
1	step (a) must always be performed before all the other steps.
2	Subsequently, all the steps follow after step (a) is performed.
Additional Information	
none	

Figure 2.3 The administrators role is to get all the pending requests to add a new PI, treat it and solve the specific PIs.

2.3.1.10 usergoal-ugMonitor

the actCoordinators 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 actCoordinators goal is to get the detailed list of existing crisis or alerts to decide on next actions to undertake.	
Reuse	
1	`ugSecurelyUseSystem [1..1]`
2	`oeGetCrisisSet [1..*]`
3	`oeGetAlertSet [1..*]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Main Steps	
a	the actor `actCoordinator` executes the `ugSecurelyUseSystem` use case
b	the actor `actCoordinator` executes the `oeGetCrisisSet` use case
c	the actor `actCoordinator` executes the `oeGetAlertSet` use case

continues in next page . . .

... Use-Case Description table continuation

Steps Ordering Constraints	
1	Step a must be executed before all other step
Additional Information	
none	

Figure 2.4 shows how coordinators monitor crisis

2.3.1.11 usergoal-ugSecurelyUseSystem

the actAdministrators 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	
Name	ugSecurelyUseSystem
Scope	system
Level	usergoal
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
the actAdministrators 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	`oeLogin [1..1]`
2	`oeLogout [1..1]`
Protocol condition(s)	
1	the iCrash system has been deployed
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the actAuthenticated is known by the system not to be logged.
Main Steps	
a	the actor `actAuthenticated` executes the `oeLogin` use case
b	the actor `actAuthenticated` executes the `oeLogout` use case
Steps Ordering Constraints	
1	step (a) must always precede step (b).
Additional Information	
none	

Figure 2.5 All the users of the system can log in and out.

2.3.1.12 subfunction-oeAddCoordinator

goal is to add a new coordinator to the system

Use-Case Description	
Name	oeAddCoordinator

continues in next page ...

. . . Use-Case Description table continuation

Scope	system
Level	subfunction
Parameters	
AdtCoordinatorID: dtCoordinatorID 1 AdtLogin: dtLogin 2 AdtPassword: dtPassword 3 CoordinatorAccessRights: etCrisisType 4	
Primary actor(s)	
1	`actAdministrator[active]`
Goal(s) description	
goal is to add a new coordinator to the system	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.13 subfunction-oeAddPI

The administrators goal is to add a new requested point of interest.

USE-CASE DESCRIPTION	
Name	oeAddPI
Scope	system
Level	subfunction
Parameters	
APIName: dtName 1 APICity: dtCity 2 APIGPSLocation: dtGPSLocation 3 APIDescription: dtDescription 4 APICategory: etCategory 5	
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actPerson[passive]`

continues in next page . . .

... Use-Case Description table continuation

Goal(s) description	
The administrators goal is to add a new requested point of interest.	
Protocol condition(s)	
1	the iCrash system has been deployed. the requested PI exists among the requests. the requested PI must have the status as solved.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the requested PI is successfully added. a message is returned to the administrator and the person that the PI has been added.
Additional Information	
none	

2.3.1.14 subfunction-oeAlert

Used by actor actComCompany to create a new alert.

USE-CASE DESCRIPTION	
Name	oeAlert
Scope	system
Level	subfunction
Parameters	
APersonType: etPersonType 1 APIName: dtName 2 APIGPSLocation: dtGPSLocation 3 AdtDate: dtDate 4 AdtTime: dtTime 5 AProblemDescription: dtDescription 6 PhoneNumber: dtPhoneNumber 7	
Primary actor(s)	
1	`actComCompany[active]`
Goal(s) description	
Used by actor actComCompany to create a new alert.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	

continues in next page ...

... Use-Case Description table continuation

Additional Information
none

2.3.1.15 subfunction-oeCheckAvailability

The coordinators goal is to check whether the requested PI exists already.

USE-CASE DESCRIPTION	
Name	oeCheckAvailability
Scope	system
Level	subfunction
Parameters	
ARequestID: dtID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actPerson[passive]`
Goal(s) description	
The coordinators goal is to check whether the requested PI exists already.	
Protocol condition(s)	
1	the iCrash system has been deployed. a request must exist in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the ignored attribute of the request is modified.
Additional Information	
none	

2.3.1.16 subfunction-oeCloseCrisis

goal is to set the status of a crisis to closed

USE-CASE DESCRIPTION	
Name	oeCloseCrisis
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to set the status of a crisis to closed	
Protocol condition(s)	
1	1

continues in next page . . .

... Use-Case Description table continuation

Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.17 subfunction-oeCreateSystemAndEnvironment

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.

USE-CASE DESCRIPTION	
Name	oeCreateSystemAndEnvironment
Scope	system
Level	subfunction
Primary actor(s)	
1	`actMsrCreator[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.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.18 subfunction-oeDeleteCoordinator

goal is to delete coordinator in the system

USE-CASE DESCRIPTION	
Name	oeDeleteCoordinator
Scope	system
Level	subfunction
Parameters	
AdtCoordinatorID: dtCoordinatorID 1	
Primary actor(s)	
1	`actAdministrator[active]`
Goal(s) description	
goal is to delete coordinator in the system	
Protocol condition(s)	
1	1

continues in next page ...

... Use-Case Description table continuation

Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.19 subfunction-oeDeletePI

The administrators goal is to delete an existing point of interest.

USE-CASE DESCRIPTION	
Name	oeDeletePI
Scope	system
Level	subfunction
Parameters	
APIID: dtID 1	
Primary actor(s)	
1	`actAdministrator[active]`
Goal(s) description	
The administrators goal is to delete an existing point of interest.	
Protocol condition(s)	
1	the iCrash system has been deployed. the to be deleted point of interest exists in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the point of interest is deleted. a message is returned to the administrator that the PI was successfully deleted.
Additional Information	
none	

2.3.1.20 subfunction-oeDeliverRequest

The coordinators goal is to change the status of the request an hand it over to the administrator.

USE-CASE DESCRIPTION	
Name	oeDeliverRequest
Scope	system
Level	subfunction
Parameters	
ARequestID: dtID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1	actPerson[passive]
2	actAdministrator[passive]
Goal(s) description	
The coordinators goal is to change the status of the request an hand it over to the administrator.	
Protocol condition(s)	
1	the iCrash system has been deployed. a request must exist in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the request status is changed to pending. the coordinator gets a message that the request has been delivered to the administrator.
Additional Information	
none	

2.3.1.21 subfunction-oeGetAlertSet

goal is to request a list with every existing alert in the system matching the given parameters

USE-CASE DESCRIPTION	
Name	oeGetAlertSet
Scope	system
Level	subfunction
Parameters	
AetAlertStatus: etAlertStatus 1	
Primary actor(s)	
1	actCoordinator[active]
Goal(s) description	
goal is to request a list with every existing alert in the system matching the given parameters	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.22 subfunction-oeGetAllRequests

The coordinators goal is to get a list of requests sent by the person.

USE-CASE DESCRIPTION	
Name	oeGetAllRequests
Scope	system
Level	subfunction

continues in next page . . .

... Use-Case Description table continuation

Primary actor(s)	
1	`actCoordinator[active]`
Secondary actor(s)	
1	`actPerson[multiple]`
Goal(s) description	
The coordinators goal is to get a list of requests sent by the person.	
Protocol condition(s)	
1	the iCrash system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the coordinator gets a list of requests sent by the person.
Additional Information	
none	

2.3.1.23 subfunction-oeGetAllRequestsFromCoordinator

The administrators goal is to retrieve pending requests.

USE-CASE DESCRIPTION	
Name	oeGetAllRequestsFromCoordinator
Scope	system
Level	subfunction
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actCoordinator[passive]`
2	`actPerson[multiple]`
Goal(s) description	
The administrators goal is to retrieve pending requests.	
Protocol condition(s)	
1	the iCrash system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the administrator gets a list of pending requests.
Additional Information	
none	

2.3.1.24 subfunction-oeGetCrisisSet

goal is to request a list with every existing crisis in the system matching the given parameters

USE-CASE DESCRIPTION	
Name	oeGetCrisisSet
Scope	system

continues in next page . . .

... Use-Case Description table continuation

Level	subfunction
Parameters	
AetCrisisStatus: etCrisisStatus 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to request a list with every existing crisis in the system matching the given parameters	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.25 subfunction-oeGetDescription

The persons goal is to get an exact description of the selected point of interest.

USE-CASE DESCRIPTION	
Name	oeGetDescription
Scope	system
Level	subfunction
Parameters	
APIID: dtID 1	
Primary actor(s)	
1	`actPerson[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
Goal(s) description	
The persons goal is to get an exact description of the selected point of interest.	
Protocol condition(s)	
1	the iCrash system has been deployed. the point of interest must figure between the PIs in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a message with the exact description of the PI is sent to the person.
Additional Information	
none	

2.3.1.26 subfunction-oeGetGPSLocation

The persons goal is to get an exact GPS location of the selected point of interest.

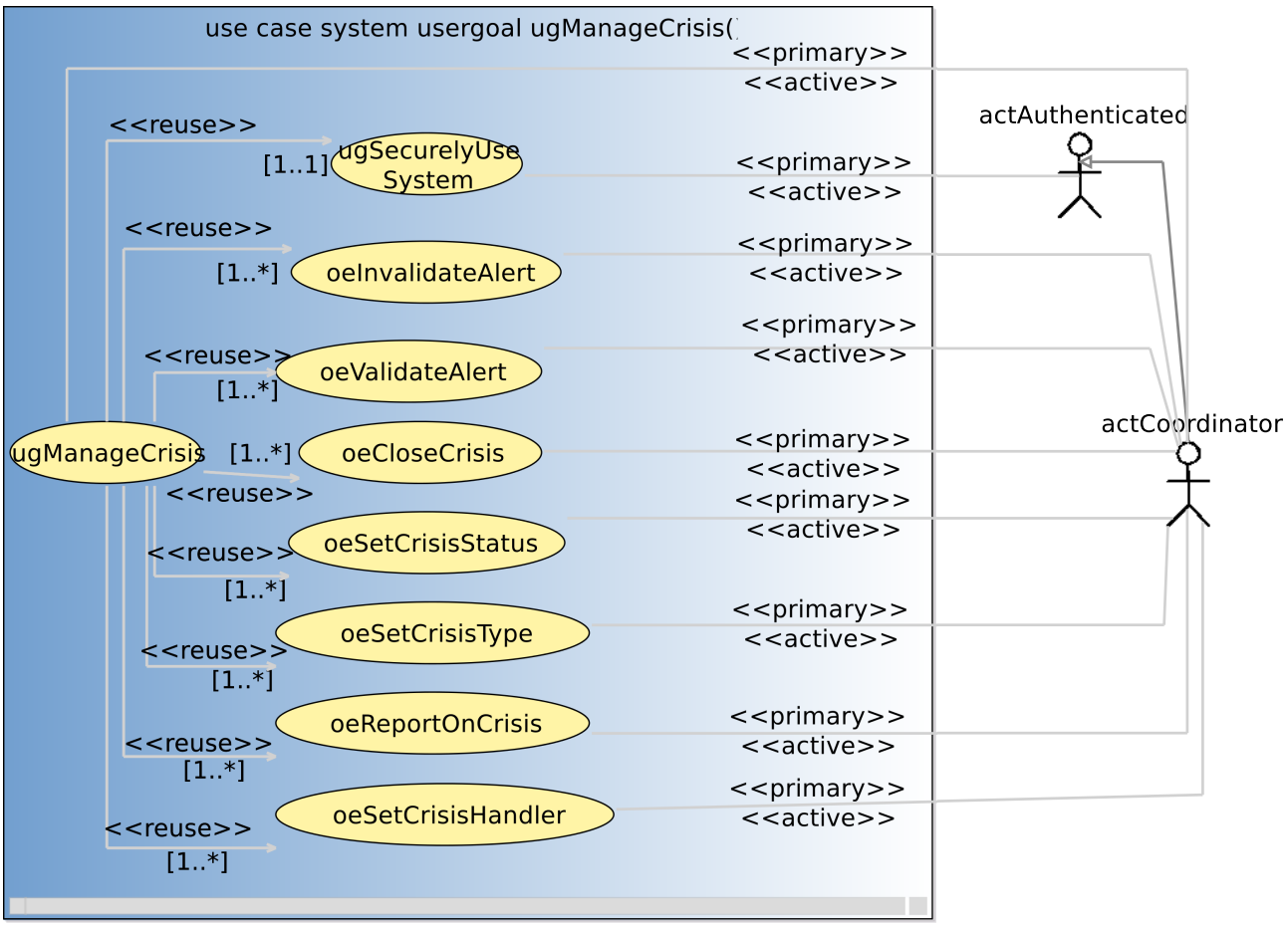

Figure 2.2: user goal Manage Crisis

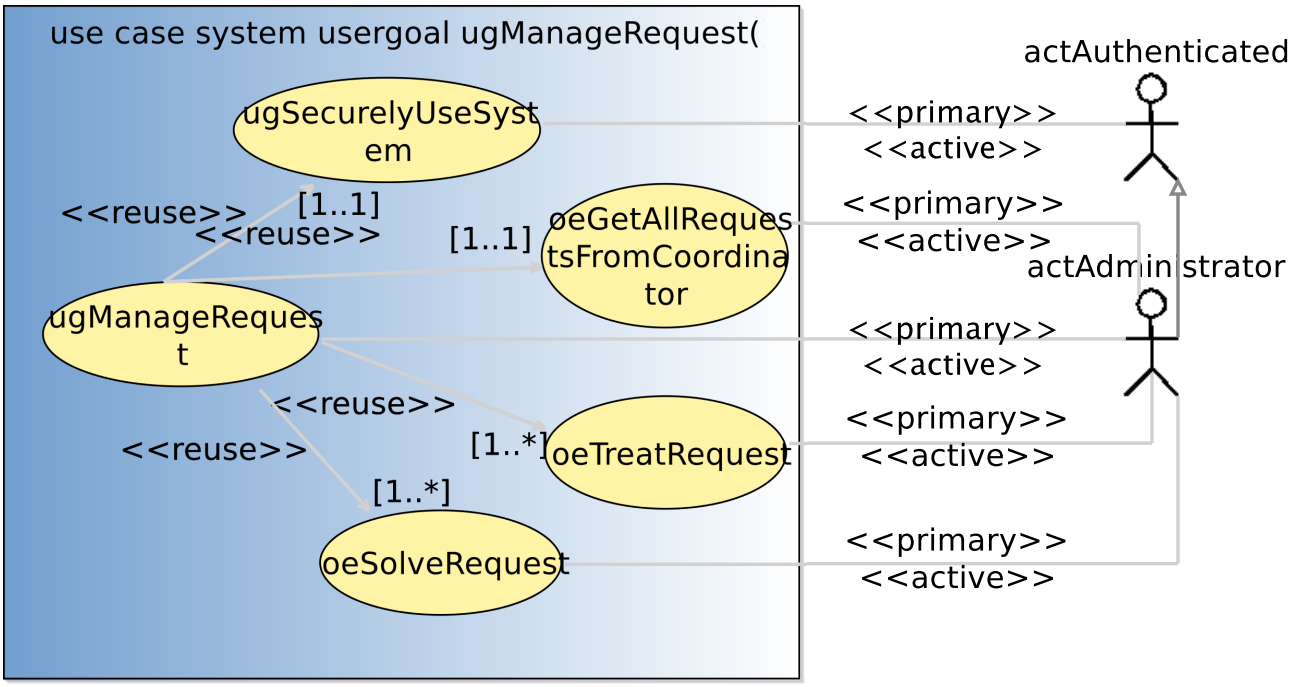

Figure 2.3: ugManageRequest user goal use case

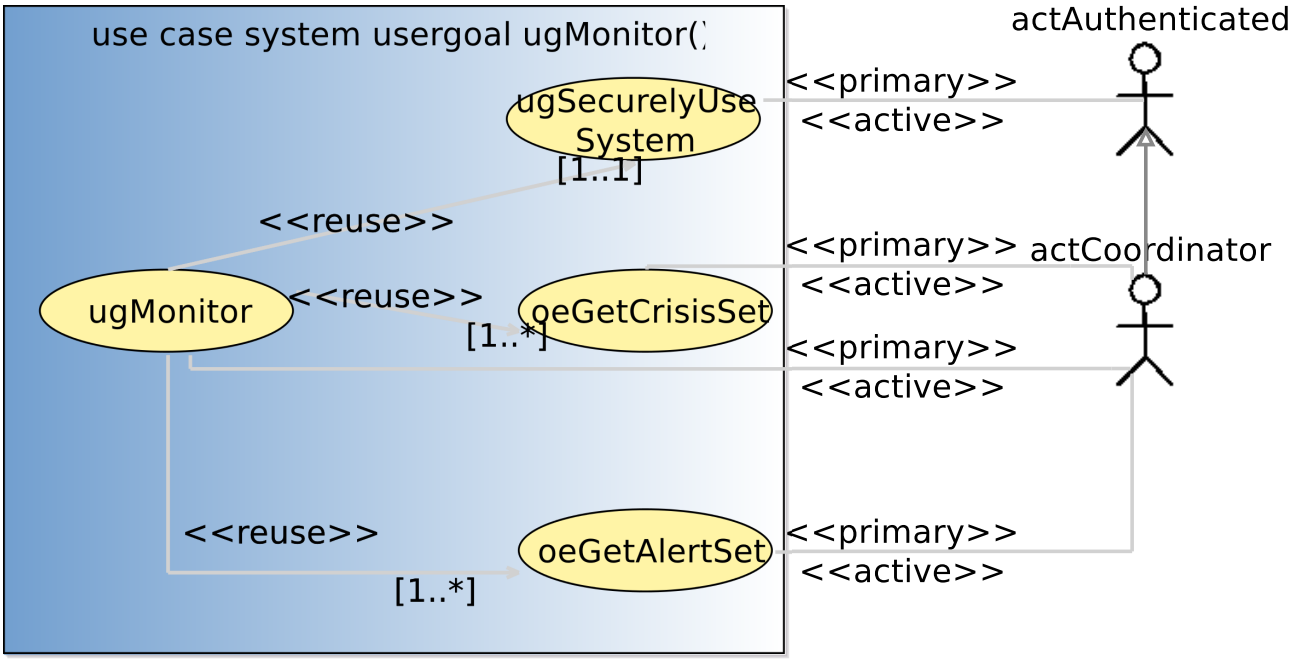

Figure 2.4: ugMonitor view

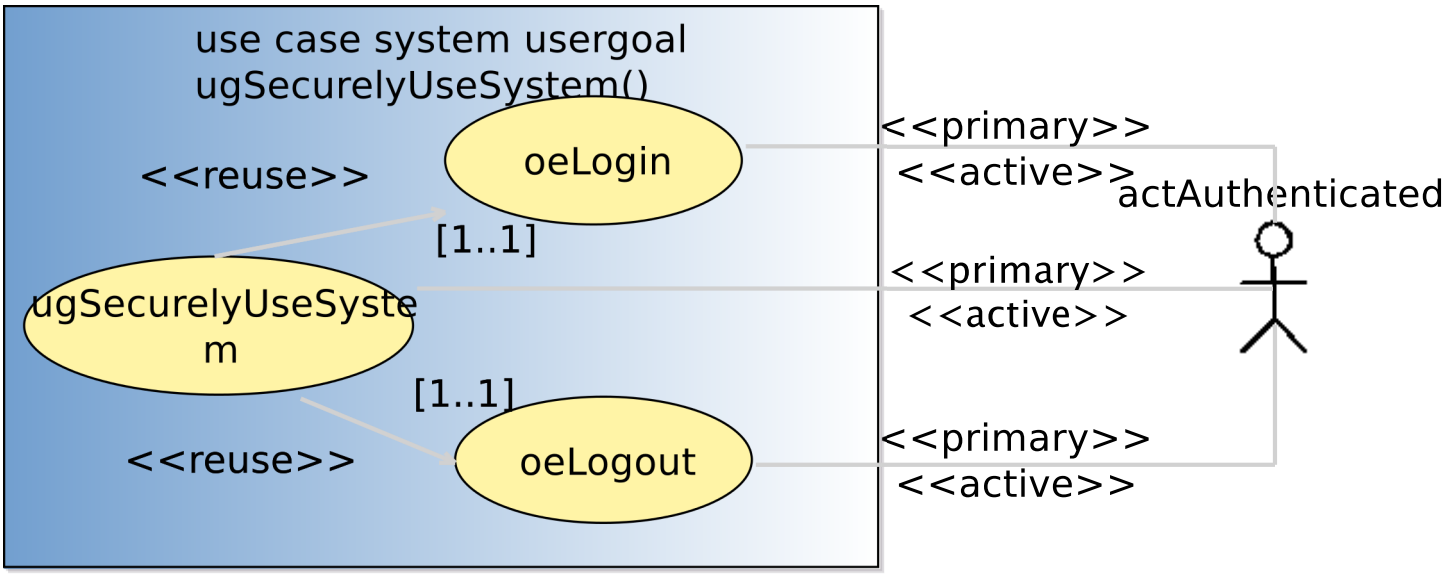

Figure 2.5: ugSecurelyUseSystem user goal use case

Use-Case Description	
Name	oeGetGPSLocation
Scope	system
Level	subfunction
Parameters	
APIID: dtID 1	
Primary actor(s)	
1	`actPerson[active]`
Secondary actor(s)	
1	`actAdministrator[passive]`
Goal(s) description	
The persons goal is to get an exact GPS location of the selected point of interest.	
Protocol condition(s)	
1	the iCrash system has been deployed. the point of interest must figure between the PIs in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a message with the exact exact GPS location of the PI is sent to the person.
Additional Information	
none	

2.3.1.27 subfunction-oeInvalidateAlert

goal is to set an alerts status to invalid

Use-Case Description	
Name	oeInvalidateAlert
Scope	system
Level	subfunction
Parameters	
AdtAlertID: dtAlertID 1	

2.3.1.28 subfunction-oeLogin

Used by any actor that is inherited by actAuthenticated to try and authenticate himself.

Use-Case Description	
Name	oeLogin
Scope	system
Level	subfunction
Parameters	
ALogin: dtLogin 1 APassword: dtPassword 2	
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
Used by any actor that is inherited by actAuthenticated to try and authenticate himself.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.29 subfunction-oeLoginWithCaptcha

To use by any actor that inherits actAuthenticate to authenticate the actor as not being a robot/machine.

Use-Case Description	
Name	oeLoginWithCaptcha
Scope	system
Level	subfunction
Parameters	
ALogin: dtLogin 1 APassword: dtPassword 2 ACaptcha: dtCaptcha 3	
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
To use by any actor that inherits actAuthenticate to authenticate the actor as not being a robot/machine.	
Protocol condition(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1
Pre-condition(s)
1
Main post-condition(s)
1
Additional Information
none

2.3.1.30 subfunction-oeLogout

Used by any actor that is inherited by actAuthenticated to logout of the system.

USE-CASE DESCRIPTION	
Name	oeLogout
Scope	system
Level	subfunction
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
Used by any actor that is inherited by actAuthenticated to logout of the system.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.31 subfunction-oeReportOnCrisis

goal is to add a report to a crisis

USE-CASE DESCRIPTION	
Name	oeReportOnCrisis
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1 AdtComment: dtComment 2	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to add a report to a crisis	

continues in next page . . .

... Use-Case Description table continuation

Protocol condition(s)
1
Pre-condition(s)
1
Main post-condition(s)
1
Additional Information
none

2.3.1.32 subfunction-oeResetPassword

Used by any actor that is inherited by actAuthenticated to receive a new password.

Use-Case Description	
Name	oeResetPassword
Scope	system
Level	subfunction
Parameters	
ALogin: dtLogin 1	
Primary actor(s)	
1	`actAuthenticated[active]`
Goal(s) description	
Used by any actor that is inherited by actAuthenticated to receive a new password.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.33 subfunction-oeSearchPI

The persons goal is to get a match by searching for a point of interest.

Use-Case Description	
Name	oeSearchPI
Scope	system
Level	subfunction
Parameters	
APIName: dtName 1	
APICategory: etCategory 2	

continues in next page . . .

. . . Use-Case Description table continuation

APICity: dtCity 3
Primary actor(s)
1 actPerson[active]
Secondary actor(s)
1 actAdministrator[passive]
Goal(s) description
The persons goal is to get a match by searching for a point of interest.
Protocol condition(s)
1 the iCrash system has been deployed.
Pre-condition(s)
1 none
Main post-condition(s)
1 a message is returned to the person that the PI has been found, or not.
Additional Information
none

2.3.1.34 subfunction-oeSendNewRequest

The persons goal is to request a new point of interest to be added to the system.

USE-CASE DESCRIPTION	
Name	oeSendNewRequest
Scope	system
Level	subfunction
Parameters	
APIName: dtName 1 APICategory: etCategory 2 APICity: dtCity 3	
Primary actor(s)	
1	actPerson[active]
Secondary actor(s)	
1	actCoordinator[passive]
Goal(s) description	
The persons goal is to request a new point of interest to be added to the system.	
Protocol condition(s)	
1	the iCrash system has been deployed.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	a new request is created. a message is returned to the person that the request has been sent.
Additional Information	
none	

2.3.1.35 subfunction-oeSetClock

Used by actActivator to change the value of the time.

Use-Case Description	
Name	oeSetClock
Scope	system
Level	subfunction
Parameters	
AcurrentTime: dtDateAndTime 1	
Primary actor(s)	
1	`actActivator[proactive]`
Goal(s) description	
Used by actActivator to change the value of the time.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.36 subfunction-oeSetCrisisHandler

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

Use-Case Description	
Name	oeSetCrisisHandler
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to declare himself as been the handler of a crisis having a specified id.	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.37 subfunction-oeSetCrisisStatus

goal is to change the status of a crisis

Use-Case Description	
Name	oeSetCrisisStatus
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1 AetCrisisStatus: etCrisisStatus 2	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to change the status of a crisis	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.38 subfunction-oeSetCrisisType

goal is to change the type of a crisis

Use-Case Description	
Name	oeSetCrisisType
Scope	system
Level	subfunction
Parameters	
AdtCrisisID: dtCrisisID 1 AetCrisisType: etCrisisType 2	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to change the type of a crisis	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	

continues in next page . . .

. . . Use-Case Description table continuation

1	1
Additional Information	
none	

2.3.1.39 subfunction-oeSollicitateCrisisHandling

the actActivators goal is to decrease the number of unhandled crisis.

USE-CASE DESCRIPTION	
Name	oeSollicitateCrisisHandling
Scope	system
Level	subfunction
Primary actor(s)	
1	`actActivator[proactive]`
Goal(s) description	
the actActivators goal is to decrease the number of unhandled crisis.	
Protocol condition(s)	
1	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.
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.1.40 subfunction-oeSolveRequest

The administrators goal is to solve a treated request.

USE-CASE DESCRIPTION	
Name	oeSolveRequest
Scope	system
Level	subfunction
Parameters	
ARequestID: dtID 1	
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actPerson[passive]`
Goal(s) description	
The administrators goal is to solve a treated request.	
Protocol condition(s)	
1	the iCrash system has been deployed. a treated request must exist in the system.

continues in next page . . .

... Use-Case Description table continuation

Pre-condition(s)	
1	none
Main post-condition(s)	
1	the request status is changed to solved. a message request solved is sent to the administrator.
Additional Information	
none	

2.3.1.41 subfunction-oeTreatRequest

The administrators goal is to treat a pending request.

USE-CASE DESCRIPTION	
Name	oeTreatRequest
Scope	system
Level	subfunction
Parameters	
ARequestID: dtID 1	
Primary actor(s)	
1	`actAdministrator[active]`
Secondary actor(s)	
1	`actPerson[passive]`
Goal(s) description	
The administrators goal is to treat a pending request.	
Protocol condition(s)	
1	the iCrash system has been deployed. a pending request must exist in the system.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the request status is changed to treated. a message request being treated is sent to the administrator.
Additional Information	
none	

2.3.1.42 subfunction-oeUpdateCoordinatorAccessRights

goal is to update a coordinators access rights in the system

USE-CASE DESCRIPTION	
Name	oeUpdateCoordinatorAccessRights
Scope	system
Level	subfunction
Parameters	
AdtCoordinatorID: dtCoordinatorID 1 CoordinatorAccessRights: etCrisisType 2	

continues in next page . . .

. . . Use-Case Description table continuation

Primary actor(s)	
1	actAdministrator[active]
Goal(s) description	
goal is to update a coordinators access rights in the system	
Protocol condition(s)	
1	1
Pre-condition(s)	
1	1
Main post-condition(s)	
1	1
Additional Information	
none	

2.3.1.43 subfunction-oeUpdatePI

The administrators goal is to update the existing point of interest information.

USE-CASE DESCRIPTION	
Name	oeUpdatePI
Scope	system
Level	subfunction
Parameters	
APIID: dtID 1 APIName: dtName 2 APICity: dtCity 3 APIGPSLocation: dtGPSLocation 4 APIDescription: dtDescription 5 APICategory: etCategory 6	
Primary actor(s)	
1	actAdministrator[active]
Goal(s) description	
The administrators goal is to update the existing point of interest information.	
Protocol condition(s)	
1	the iCrash system has been deployed. there exists already the point of interest which has to be updated.
Pre-condition(s)	
1	none
Main post-condition(s)	
1	the point of interest has its new values. a message is returned to the administrator that the PI is up to date.

continues in next page . . .

. . . Use-Case Description table continuation

Additional Information
none

2.3.1.44 subfunction-oeValidateAlert

goal is to set an alerts status to valid

USE-CASE DESCRIPTION	
Name	oeValidateAlert
Scope	system
Level	subfunction
Parameters	
AdtAlertID: dtAlertID 1	
Primary actor(s)	
1	`actCoordinator[active]`
Goal(s) description	
goal is to set an alerts status to valid	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	
Additional Information	
none	

2.3.2 Use Case Instance(s)

2.3.2.1 Use-Case Instance - ucisuAddNewPI:suAddNewPI

Represents the instance of adding a new point of interest to the system.

SUMMARY USE-CASE INSTANCE
Instantiated Use Case suAddNewPI
Instance ID ucisuAddNewPI

Figure 2.6 The person searches for a specific PI which is not in the system. The coordinator checks if the problem is really true and sends it to the administrator. The administrator treats the request and adds the new PI to the system.

2.3.2.2 Use-Case Instance - ucisuGenerateNewAlert:suGenerateNewAlert

Represents the instance of alert the system of a potential crisis.

SUMMARY USE-CASE INSTANCE
Instantiated Use Case suGenerateNewAlert
Instance ID ucisuGenerateNewAlert

Figure 2.7 The person sends a Sms to the communication company about a possible alert to handle.

2.3.2.3 Use-Case Instance - ucisuGlobalCrisisHandling:suGlobalCrisisHandling

shows how the coordinator handles crisis

SUMMARY USE-CASE INSTANCE
Instantiated Use Case suGlobalCrisisHandling
Instance ID ucisuGlobalCrisisHandling

Figure 2.8 an example of how crisis are handled globally

2.3.2.4 Use-Case Instance - ucisuScenarioPresentation:suScenarioPresentation

Represents the instance of the three combined variants.

SUMMARY USE-CASE INSTANCE
Instantiated Use Case suScenarioPresentation

continues in next page . . .

. . . summary Use-Case Instance table continuation

Instance ID
ucisuScenarioPresentation

Figure 2.9 Present the collaboration of the three variants.

2.3.2.5 Use-Case Instance - uciugAdministrateTheSystem:ugAdministrateTheSystem

A concrete instance how you can administrate the system.

USERGOAL USE-CASE INSTANCE
Instantiated Use Case
ugAdministrateTheSystem
Instance ID
uciugAdministrateTheSystem

Figure 2.10 example of how an administrator creates new coordinators

2.3.2.6 Use-Case Instance - uciugLoginWithCaptcha:ugLoginWithCaptcha

A concrete scenario how a coordinator fails to login three times in a row and has to solve the captcha.

USERGOAL USE-CASE INSTANCE
Instantiated Use Case
ugLoginWithCaptcha
Instance ID
uciugLoginWithCaptcha

Figure 2.11 Logging in using the captcha. You first need to fail to login three times in a row.

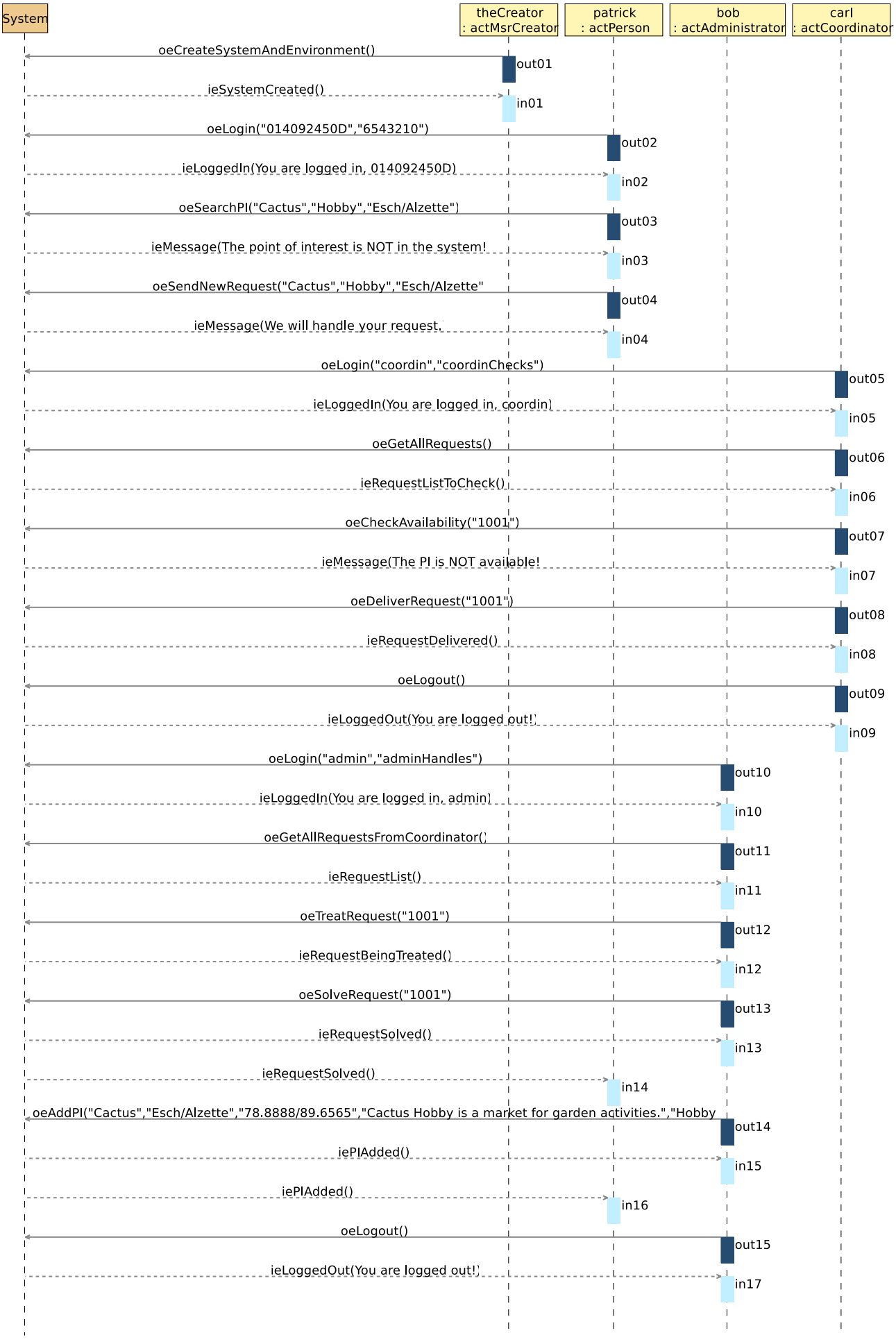
System
theCreator : actMsrCreator
patrick : actPerson
bob : actAdministrator
carl : actCoordinator
oeCreateSystemAndEnvironment()
out01
ieSystemCreated()
in01
oeLogin("014092450D","6543210")
out02
ieLoggedIn(You are logged in, 014092450D)
in02
oeSearchPI("Cactus","Hobby","Esch/Alzette")
out03
ieMessage(The point of interest is NOT in the system!
in03
oeSendNewRequest("Cactus","Hobby","Esch/Alzette"
out04
ieMessage(We will handle your request.
in04
oeLogin("coordin","coordinChecks")
out05
ieLoggedIn(You are logged in, coordin)
in05
oeGetAllRequests()
out06
ieRequestListToCheck()
in06
oeCheckAvailability("1001")
out07
ieMessage(The PI is NOT available!
in07
oeDeliverRequest("1001")
out08
ieRequestDelivered()
in08
oeLogout()
out09
ieLoggedOut(You are logged out!)
in09
oeLogin("admin","adminHandles")
out10
ieLoggedIn(You are logged in, admin)
in10
oeGetAllRequestsFromCoordinator()
out11
ieRequestList()
in11
oeTreatRequest("1001")
out12
ieRequestBeingTreated()
in12
oeSolveRequest("1001")
out13
ieRequestSolved()
in13
ieRequestSolved()
in14
oeAddPI("Cactus","Esch/Alzette","78.8888/89.6565","Cactus Hobby is a market for garden activities.","Hobby
out14
iePIAdded()
in15
iePIAdded()
in16
oeLogout()
out15
ieLoggedOut(You are logged out!)
in17

Figure 2.7: suGenerateNewAlert summary use case instance


System
bob
: actCoordinator
steve
: actCoordinator
oeLogin("bob","password")
out01
ieLogin(You successfully logged in!)
in01
oeLogin("steve","password")
out02
ieLogin(You successfully logged in!)
in02
oeGetAlertSet("pending","2")
out03
ieMessage(AlertSet)
in03
oeValidateAlert("1")
out04
ieMessage(Permission denied! Your access rights are to low to validate this alert
in04
oeGetAlertSet("pending","2")
out05
ieMessage(AlertSet)
in05
oeValidateAlert("1")
out06
ieMessage(Alert validated)
in06
oeGetCrisisSet("pending","2")
out07
ieMessage(CrisisSet)
in07
oeSetCrisisHandler("1")
out08
ieMessage(Permission denied! Your access rights are to low to handle this crisis
in08
oeGetCrisisSet("pending","1")
out09
ieMessage(CrisisSet)
in09
oeSetCrisisHandler("1")
out10
ieMessage(Crisis handler set to steve
in10
oeSetCrisisStatus("1","in progression")
out11
ieMessage(Crisis status set to in progression)
in11
oeReportOnCrisis("1","Crisis handled")
out12
ieMessage(Report added to the crisis
in12
oeCloseCrisis("1")
out13
ieMessage(Crisis closed)
in13
oeSetCrisisType("2","1")
out14
ieMessage(Crisis type set to 1)
in14
oeGetCrisisSet("pending","1")
out15
ieMessage(CrisisSet)
in15
oeSetCrisisHandler("2")
out16
ieMessage(Crisis handler set to bob)
in16
oeSetCrisisStatus("2","in progression")
out17
ieMessage(Crisis status set to in progression)
in17
oeReportOnCrisis("2","Crisis handled")
out18
ieMessage(Report added to the crisis
in18
oeCloseCrisis("2")
out19
ieMessage(Crisis closed)
in19
oeLogout()
out20
ieLogout(You successfully logged out!)
in20
oeLogout()
out21
ieLogout(You successfully logged out!)
in21

Figure 2.8:

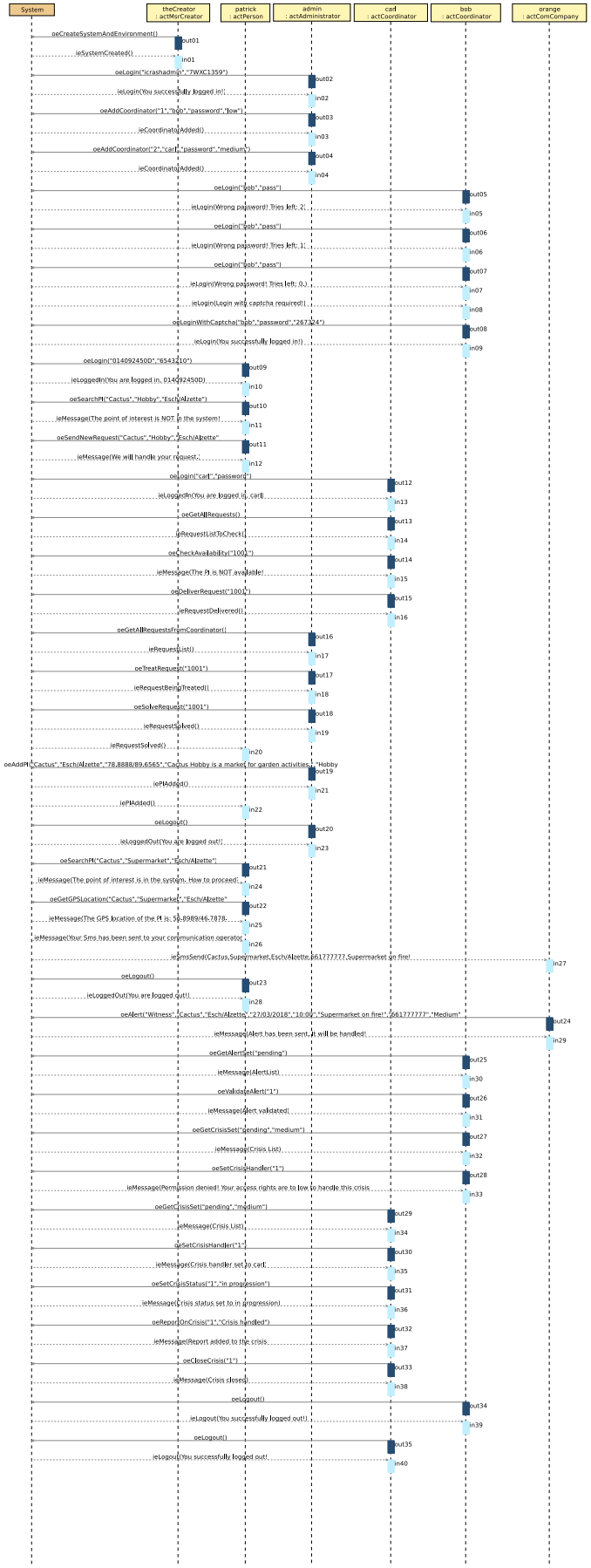

Figure 2.9: uci-suScenarioPresentation-uciSceanrioPresentation use case instance sequence diagram

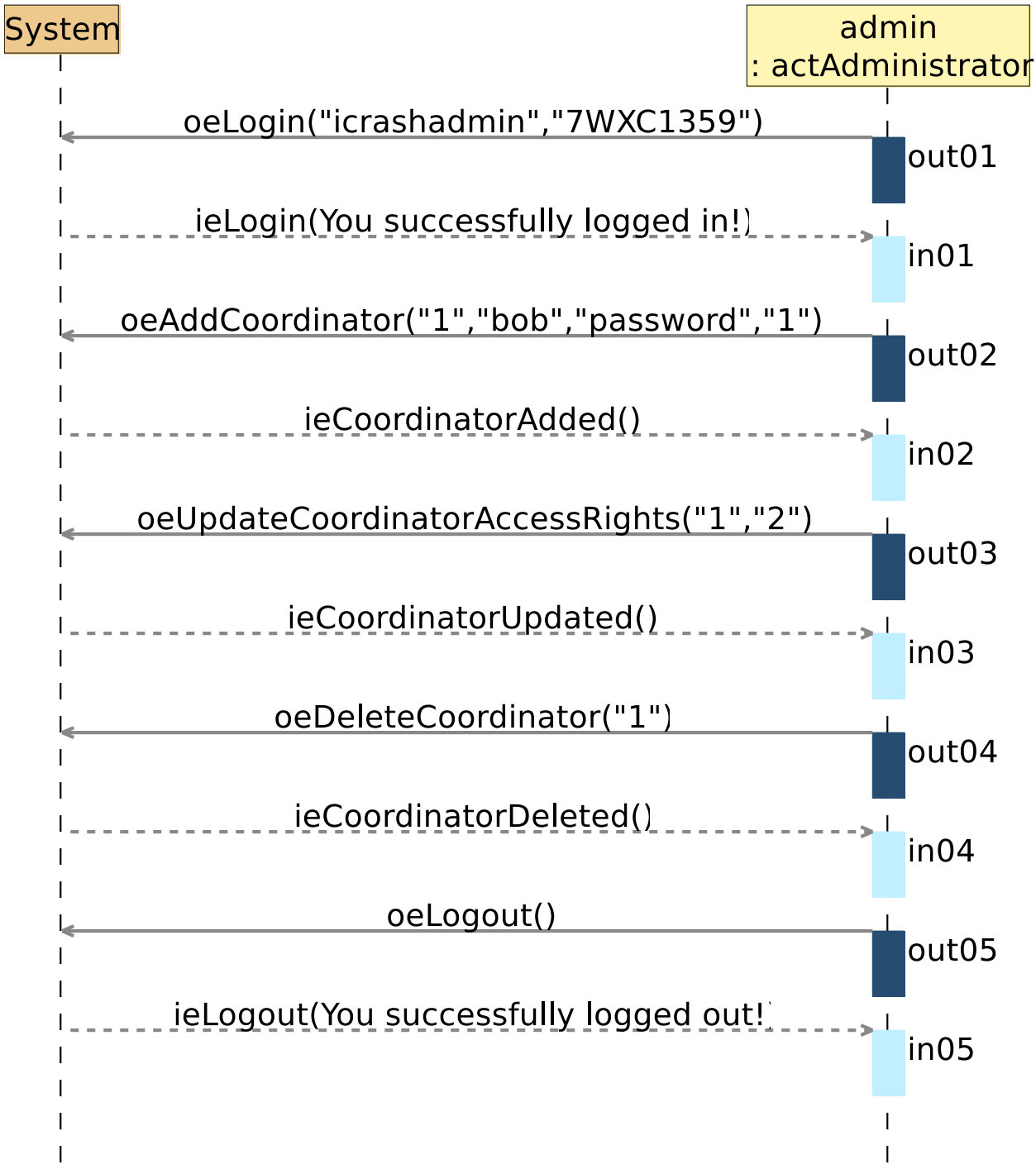
System
admin
: actAdministrator
oeLogin("icrashadmin","7WXC1359")
out01
ieLogin(You successfully logged in!)
in01
oeAddCoordinator("1","bob","password","1")
out02
ieCoordinatorAdded()
in02
oeUpdateCoordinatorAccessRights("1","2")
out03
ieCoordinatorUpdated()
in03
oeDeleteCoordinator("1")
out04
ieCoordinatorDeleted()
in04
oeLogout()
out05
ieLogout(You successfully logged out!)
in05

Figure 2.10:


System
bob
: actAuthenticated
oeLogin("bob","qwertz")
out01
ieLogin(Wrong Password, try again.)
in01
oeLogin("bob","idontknowanymore")
out02
ieLogin(Wrong Password, try again.)
in02
oeLogin("bob","nothing")
out03
ieLogin(Wrong Password, try again.)
in03
ieLogin(Login with captcha required.)
in04
oeLoginWithCaptcha("bob","sendhelp","C4p7cha")
out04
ieLogin(Welcome bob.)
in05

Figure 2.11:

Chapter 3

Environment Model

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

3.1 Global view 04

Figure 3.1 presents the actors in relation to the ctState.

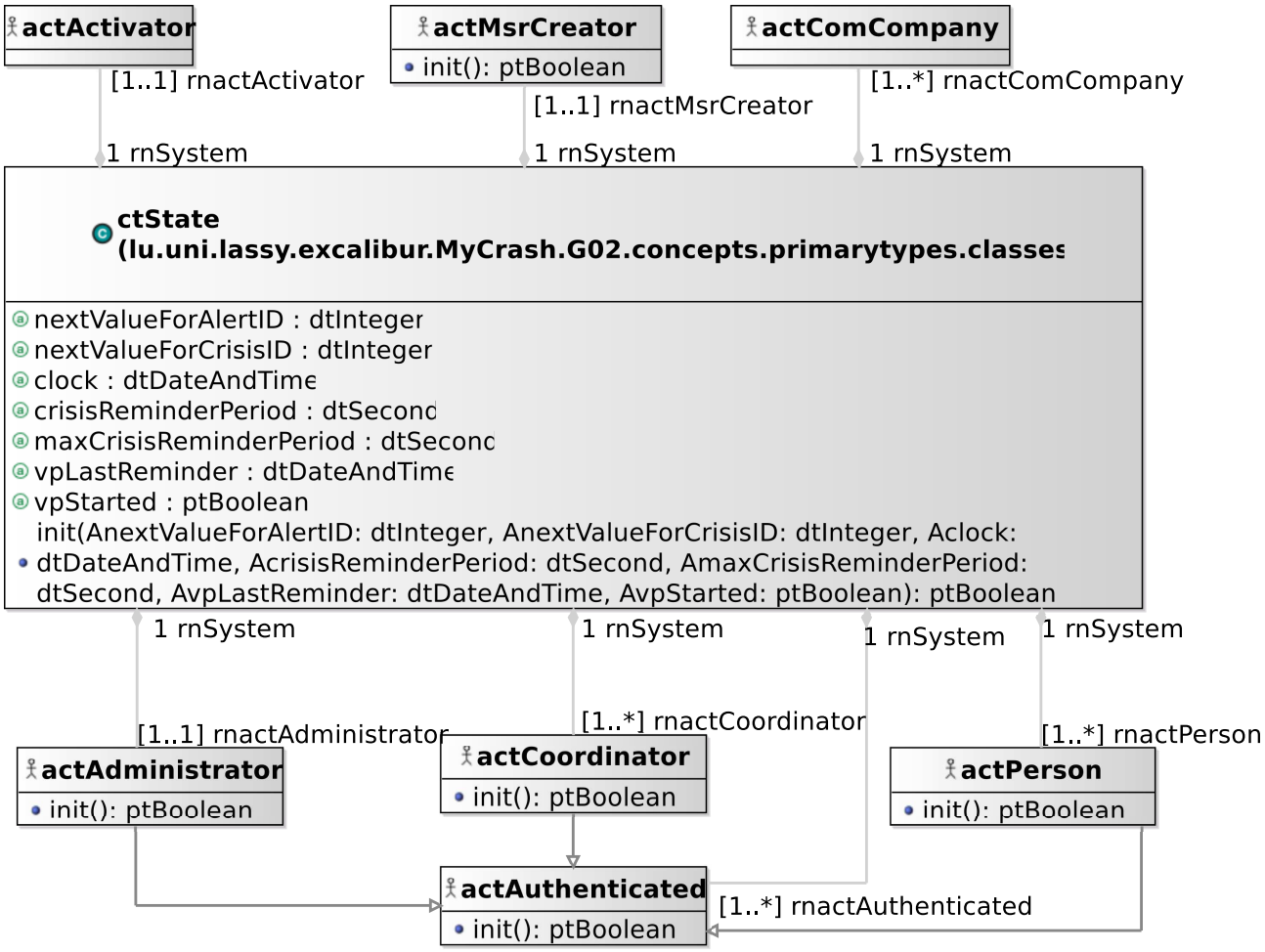

Figure 3.1: Environment Model - Global View 04. Environment global view 01.

3.2 Actors and Interfaces Descriptions

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

3.2.1 `actActivator` Actor

Actor	
actActivator	
represents a logical actor for time automatic message sending based on systems or environment status.	
OutputInterfaces	
OUT 1	**`oeSollicitateCrisisHandling():ptBoolean`** The output event oeSollicitateCrisisHandling reminds the coordinators that there are still pending crisis.

continues in next page . . .

. . . Actor table continuation

OUT 2	**oeSetClock(AcurrentTime:dtDateAndTime):ptBoolean** The output event oeSetClock updates the current time.

3.2.2 actAdministrator Actor

Actor	
actAdministrator PI variant: The actor Administrator is in charge of treating requested points of interests and adding, updating or deleting points of interests. The Administrator receives a request to add a new PI from a Person which the actor Coordinator checks before delivering them to the actor Administrator. Access Rights variant: The actor Administrator is in charge of managing coordinators. This includes creating and deleting them and updating their access rights.	
Extends	
lu.uni.lassy.excalibur.MyCrash.G02.environment.actAuthenticated	
OutputInterfaces	
OUT 1	**oeGetAllRequestsFromCoordinator():ptBoolean** The output event oeGetAllRequestsFromCoordinator retrieves a list of requests which the actor Coordinator checked.
OUT 2	**oeTreatRequest(ARequestID:dtID):ptBoolean** The output event oeTreatRequest modifies the request status to treated.
OUT 3	**oeSolveRequest(ARequestID:dtID):ptBoolean** The output event oeTreatRequest modifies the request status to solved.
OUT 4	**oeAddPI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory):ptBoolean** The output event oeAddPI adds a new point of interest requested by the actor Person.
OUT 5	**oeUpdatePI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory):ptBoolean** The output event oeUpdatePI updates a point of interest.
OUT 6	**oeDeletePI(APIID:dtID):ptBoolean** The output event oeDeletePI deletes a point of interest.
OUT 7	**oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin, AdtPassword:dtPassword, CoordinatorAccessRights:etCrisisTy** The output event oeAddCoordinator creates a new coordinator in the system.
OUT 8	**oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID):ptBoolean** The output event oeDeleteCoordinator deletes a coordinator in the system.
OUT 9	**oeUpdateCoordinatorAccessRights(AdtCoordinatorID:dtCoordinatorID, CoordinatorAccessRights:etCrisisType):ptBoolean** The output event oeUpdateCoordinatorAccessRights updates a coordinators access rights in the system.
InputInterfaces	
IN 1	**ieRequestList():ptBoolean** The input event ieRequestList returns a list of checked requests from the actor Coordinator to the actor Administrator.
IN 2	**ieRequestBeingTreated():ptBoolean** The input event ieRequestBeingTreated specifies that the request is being treated.

continues in next page . . .

... Actor table continuation

IN 3	**ieRequestSolved():ptBoolean** The input event ieRequestSolved confirms that the request was solved.
IN 4	**iePIAdded():ptBoolean** The input event iePIAdded confirms that the requested point of interest has been added.
IN 5	**iePIUpToDate():ptBoolean** The input event iePIUpToDate confirms that the requested point of interest has been updated.
IN 6	**iePIDeleted():ptBoolean** The input event iePIDeleted confirms that the requested point of interest has been deleted.
IN 7	**ieCoordinatorAdded():ptBoolean** The input event ieCoordinatorAdded confirms if a coordinator was successfully added to the system.
IN 8	**ieCoordinatorDeleted():ptBoolean** The input event ieCoordinatorDeleted confirms if a coordinator was successfully deleted in the system.
IN 9	**ieCoordinatorUpdated():ptBoolean** The input event ieCoordinatorUpdated confirms if a coordinators access rights were successfully updated in the system.

3.2.3 `actAuthenticated` Actor

Actor	
actAuthenticated	
Authentication variant: The abstract actor authenticated is getting implemented by multiple other actors. It is needed to perform simple task, like login/logout. You can also request to reset your password.	
OutputInterfaces	
OUT 1	**oeLogin(ALogin:dtLogin, APassword:dtPassword):ptBoolean** The output event oeLogin sends the username and password to the system, used to authenticate the user trying to login.
OUT 2	**oeLogout():ptBoolean** The output event oeLogout sends a logout request, from the currently logged in user, to the system.
OUT 3	**oeLoginWithCaptcha(ALogin:dtLogin, APassword:dtPassword, ACaptcha:dtCaptcha):ptBoolean** The output event oeLoginWithCaptcha sends the username, password and captcha to the system, used to authenticate the user and to confirm that the user is not a robot.
OUT 4	**oeResetPassword(ALogin:dtLogin):ptBoolean** The output event oeResetPassword requests a password request of any given user.
InputInterfaces	
IN 1	**ieLogin(AMessage:dtMessage):ptBoolean** The returned message from a positive or negative login. This input field is used when a use tries to login in any form. (Captcha or normal login)
IN 2	**ieLoggedOut(AMessage:dtMessage):ptBoolean** The input event ieLoggedOut confirms that the logout has been completed.
IN 3	**ieMessage(AMessage:dtMessage):ptBoolean** The input event ieMessage returns a message from the system that tells you useful things.

continues in next page ...

... Actor table continuation

IN 4	**ieResetPassword():ptBoolean** The input event ieResetPassword specifies that the request is being treated. Also tells you what you need to provide next.

3.2.4 actComCompany Actor

ACTOR	
actComCompany represents the communication company stakeholder ensuring the input/output of textual messages with humans having communication devices.	
OutputInterfaces	
OUT 1	**oeAlert(APersonType:etPersonType, APIName:dtName, APIGPSLocation:dtGPSLocation, AdtDate:dtDate, AdtTime:dtTime, AProblemDescription:dtDescription, PhoneNumber:dtPhoneNumber):ptBoolean** The input event oeAlert redirects an alert received from a person to the coordinators.
InputInterfaces	
IN 1	**ieSmsSend(APIName:dtName, APICategory:etCategory, ACity:dtCity, AdtPhoneNumber:dtPhoneNumber, AdtSMS:dtSMS):ptBoolean** The input event ieSmsSend returns a confirmation to the person who sent an alert.

3.2.5 actCoordinator Actor

ACTOR	
actCoordinator PI variant: The actor Coordinator is in charge of checking the requested points of interest of their existence in the system and to deliver the non-existing requested points of interest to the actor Administrator. Access Rights variant: The actor Coordinator is in charge of managing and monitoring alerts and crisis. This means he has the overview of over all alerts and crisis. He decides weather a alert should be validated or not. All the crisis managed by him, which means he is the only actor interacting and handling crisis.	
Extends	
lu.uni.lassy.excalibur.MyCrash.G02.environment.actAuthenticated	
OutputInterfaces	
OUT 1	**oeGetAllRequests():ptBoolean** The output event oeGetAllRequests retrieves a list of requests sent by the actor Person.
OUT 2	**oeCheckAvailability(ARequestID:dtID):ptBoolean** The output event oeCheckAvailability confirms that a requested point of interest sent by the actor Person exists already in the system or not and sets the request ignored attribute to the corresponding value.
OUT 3	**oeDeliverRequest(ARequestID:dtID):ptBoolean** The output event oeDeliverRequest sends a checked request sent by the actor Person to the actor Administrator.
OUT 4	**oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean** The output event oeInvalidateAlert puts the status of an alert to invalid.
OUT 5	**oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean** The output event oeValidateAlert puts the status of an alert to valid.

continues in next page ...

... Actor table continuation

OUT 6	**oeGetAlertSet(AetAlertStatus:etAlertStatus):ptBoolean** The output event oeGetAlertSet requests a list with every existing alert in the system matching the given parameters.
OUT 7	**oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean** The output event oeCloseCrisis puts the status of a crisis to closed.
OUT 8	**oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean** The output event oeGetCrisisSet requests a list with every existing crisis in the system matching the given parameters.
OUT 9	**oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean** The output event oeSetCrisisHandler sets a coordinator as handler of a crisis.
OUT 10	**oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoolea** The output event oeReportOnCrisis adds a report to a crisis.
OUT 11	**oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisStatu** The output event oeSetCrisisStatus changes the status of a crisis.
OUT 12	**oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType):ptB** The output event oeSetCrisisType changes the type of a crisis.
InputInterfaces	
IN 1	**ieRequestListToCheck():ptBoolean** The input event ieRequestListToCheck returns a list of requests which have to be checked from the actor Person to the actor Coordinator.
IN 2	**ieRequestDelivered():ptBoolean** The input event ieRequestDelivered specifies that the request has been delivered to the actor Coordinator.
IN 3	**ieCrisisSet():ptBoolean** The input event ieCrisisSet returns a list with every existing crises in the system matching the given parameters.
IN 4	**ieAlertSet():ptBoolean** The input event ieAlertSet returns a list with every existing alert in the system matching the given parameters.

3.2.6 `actMsrCreator` Actor

ACTOR
actMsrCreator Represents the creator stakeholder in charge of state and environment initialization.
OutputInterfaces
OUT 1 **oeCreateSystemAndEnvironment():ptBoolean** sent to request the initialization of the systems class instances and the environment actors instances.
InputInterfaces
IN 1 **ieSystemCreated():ptBoolean** confirmation message that the system was successfully created.

3.2.7 `actPerson` Actor

ACTOR

continues in next page ...

. . . Actor table continuation

actPerson
PI variant: The actor Person is in charge of searching for a point of interest, get the exact GPS location and send a Sms to the communication company with a specific problem description. If the the PI is not available, the actor Person has the possibility to send a request to the system to add a new PI.
Extends
lu.uni.lassy.excalibur.MyCrash.G02.environment.actAuthenticated
OutputInterfaces
OUT 1 **oeSearchPI(APIName:dtName, APICategory:etCategory, APICity:dtCity):ptBoolean** The output event oeSearchPI returns a point of interest.
OUT 2 **oeSendNewRequest(APIID:dtID, APIName:dtName, APICategory:etCategory, APICity:dtCity):ptBoolean** The output event oeSendNewRequest adds a new requested PI to the system.
OUT 3 **oeGetGPSLocation(APIID:dtID):ptBoolean** The output event oeGetGPSLocation retrieves the exact GPS location of a searched PI.
OUT 4 **oeGetDescription(APIID:dtID):ptBoolean** The output event oeGetDescription retrieves a description of a searched PI.
InputInterfaces
IN 1 **iePIAdded():ptBoolean** The input event iePIAdded confirms that the requested point of interest has been added to the system.

Chapter 4

Concept Model

4.1 PrimaryTypes-Classes

4.1.1 Local view 01

Figure 4.1 presents the relations between all the classes.

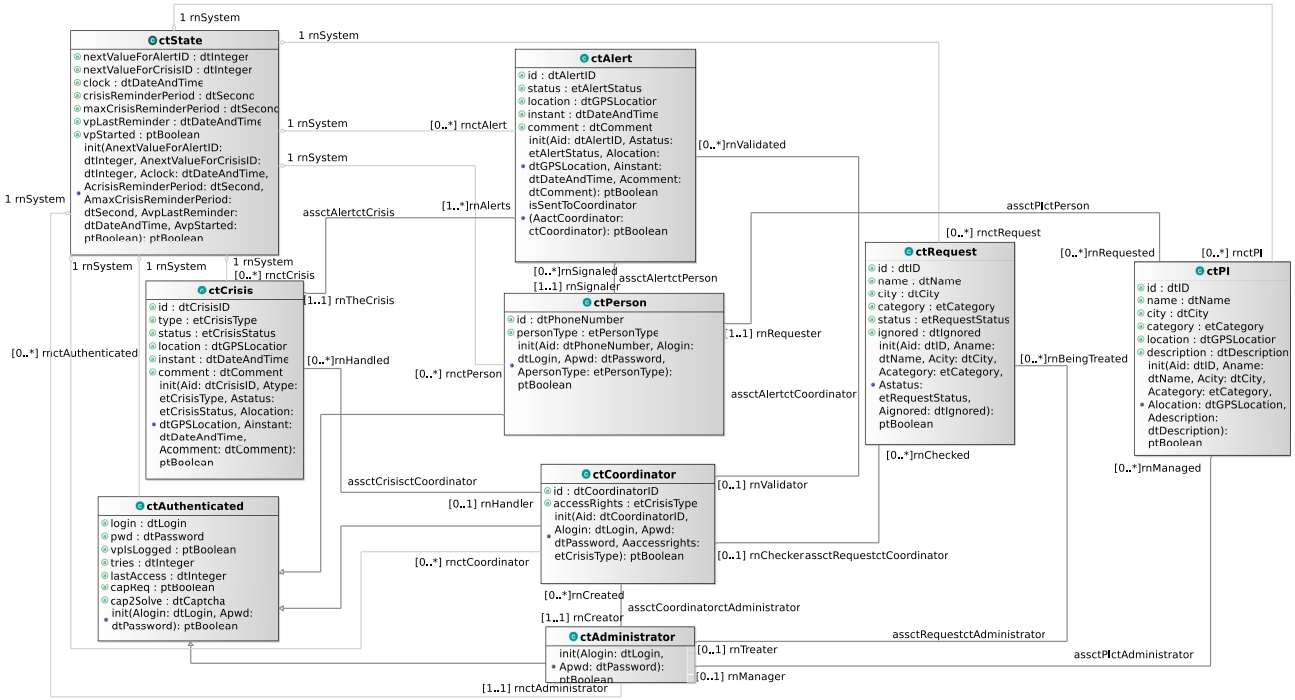

Figure 4.1: Concept Model - PrimaryTypes-Classes local view 01. PrimaryTypes Classes local view 01.

4.1.2 Global view 05

Figure 4.2 Represents class-actor relation for the PI variant

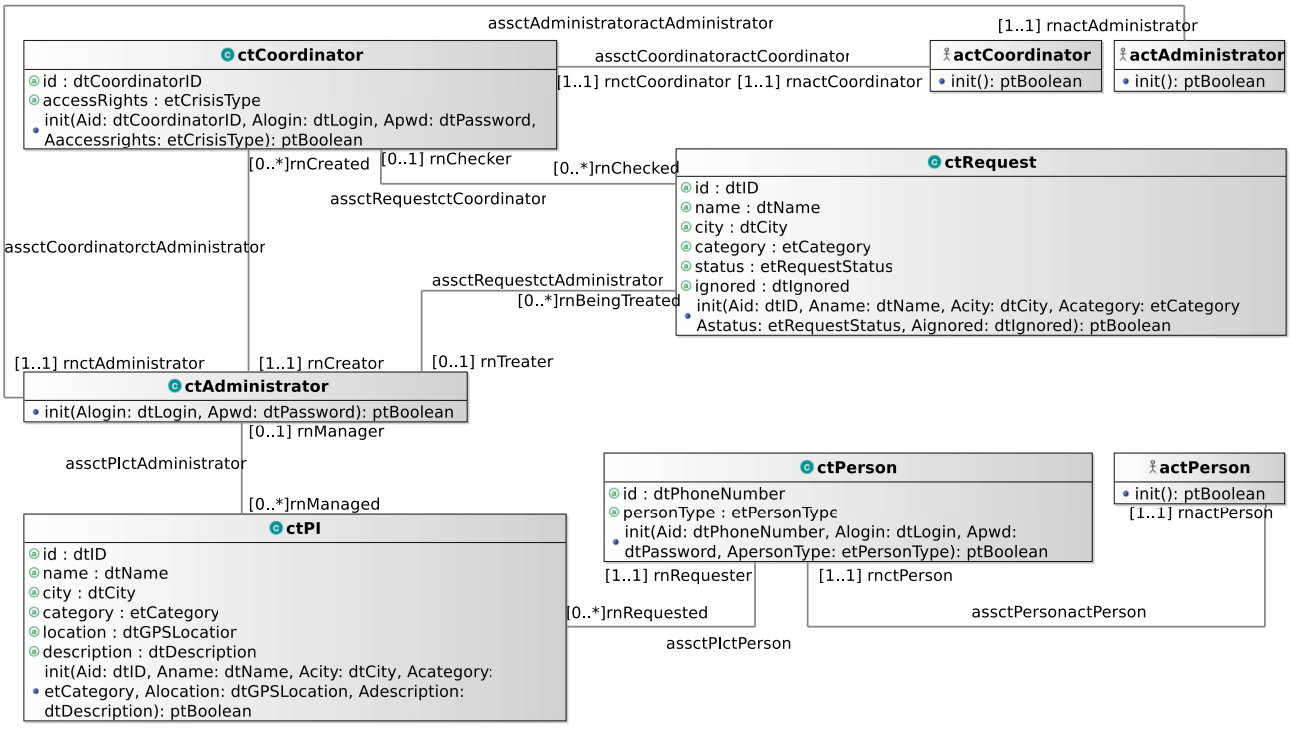

Figure 4.2: Concept Model - PrimaryTypes-Classes global view 05. PIvariant-CGV-01 concept global view 01.

4.1.3 Global view 06

Figure 4.3 Represents class and actor relation for Access Rights variant.

4.2 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.

4.2.1 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	
ctAdministrator	
used to caracterize internally the entity that is responsible of administrating the iCrash system.	
extends	lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes.ctAuthenticated
operation	**`init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean`** used to initialize the current object as a new instance of the ctAdministrator type.
ctAlert	
Used to model crisis alerts sent by any human having communication capability using communication companies belonging to the systems environment	

continues in next page . . .

. . . Classes table continuation

attribute	**comment: dtComment** a textual description providing unstructured information on the alert.
attribute	**id: dtAlertID** the alert unique identification information.
attribute	**instant: dtDateAndTime** the date and time at which the alert notification has been sent.
attribute	**location: dtGPSLocation** the position of the alert provided by the space-based satellite navigation system used by the human using the communication company to inform the iCrash system of a crisis.
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:ctCoordinator):ptBoolean** used to provide a given coordinator with current alert information.
ctAuthenticated	
used to model systems representation about actors that need to authenticate to access some specific functionalities.	
attribute	**capReq: ptBoolean** used to determine if the actor needs to solve a captcha.
attribute	**lastAccess: dtInteger** used to determine the last time the actor tried to login.
attribute	**login: dtLogin** an identifier for authentication.
attribute	**pwd: dtPassword** a key for authentication.
attribute	**tries: dtInteger** used to determine how many tries the actor tried to login.
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 systems representation about the actors that have the responsibility to handle alerts and crisis.	
extends	lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes.ctAuthenticated
attribute	**id: dtCoordinatorID** a unique identification information.
operation	**init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword, Aaccessrights:etCrisisType):ptBoolean** used to initialize the current object as a new instance of the ctCoordinator type.
ctCrisis	
Used to model crisis that are infered from the reception of at least one alert message. Crisis are entities that are handled by the iCrash system.	
attribute	**comment: dtComment** a textual description providing unstructured information on the crisis handling.

continues in next page . . .

... Classes table continuation

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 by 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	**init(Aid:dtCrisisID, Atype:etCrisisType, Astatus:etCrisisStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean** used to initialize the current object as a new instance of the ctAlert type.
ctPerson	
Used to model systems representation about the indirect actors that has alerted of potential crisis, but also perform requests for new potential points of interest.	
extends	lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes.ctAuthenticated
attribute	**id: dtPhoneNumber** the persons unique identification information of type dtPhoneNumber.
attribute	**personType: etPersonType** the person type.
operation	**init(Aid:dtPhoneNumber, Alogin:dtLogin, Apwd:dtPassword, ApersonType:etPersonType):ptBoolean** used to initialise the current object as a new instance of the ctPerson type.
ctPI	
Used to model points of interest that are introduced from the reception of at least one request. PIs are handled by the actor Administrator.	
attribute	**category: etCategory** classifies the PI in a sort of groups.
attribute	**city: dtCity** represents the location at the specific point of interest.
attribute	**description: dtDescription** a textual description providing information of the point of interest.
attribute	**id: dtID** the PIs unique identification information.
attribute	**location: dtGPSLocation** represents the location of the PI as GPS coordinates.
attribute	**name: dtName** represents the name of a point of interest.
operation	**init(Aid:dtID, Aname:dtName, Acity:dtCity, Acategory:etCategory, Alocation:dtGPSLocation, Adescription:dtDescription):ptBoolean** used to initialise the current object as a new instance of the ctPI type.
ctRequest	
Used to model requests that are introduced from the actor Person as a new potential point of interest.	
attribute	**category: etCategory**

continues in next page ...

. . . Classes table continuation

	classifies the new potential PI in a sort of groups.
attribute	**city: dtCity** represents the location at the new potential point of interest.
attribute	**id: dtID** the request's unique identification information.
attribute	**ignored: dtIgnored** represents the fact that the new to add point of interest already exists or not. Shall be ignored if yes.
attribute	**name: dtName** represents the potential name of the new point of interest.
attribute	**status: etRequestStatus** the request handling status.
operation	**init(Aid:dtID, Aname:dtName, Acity:dtCity, Acategory:etCategory, Astatus:etRequestStatus, Aignored:dtIgnored):ptBoolean** used to initialise the current object as a new instance of the ctRequest type.
ctState used to model the system. Each system specified using Messir 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 ramdomly 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 idenitification value.
attribute	**nextValueForCrisisID: dtInteger** used to associate each crisis declared with a unique idenitification 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** used to initialize the current object as a new instance of the ctState type.

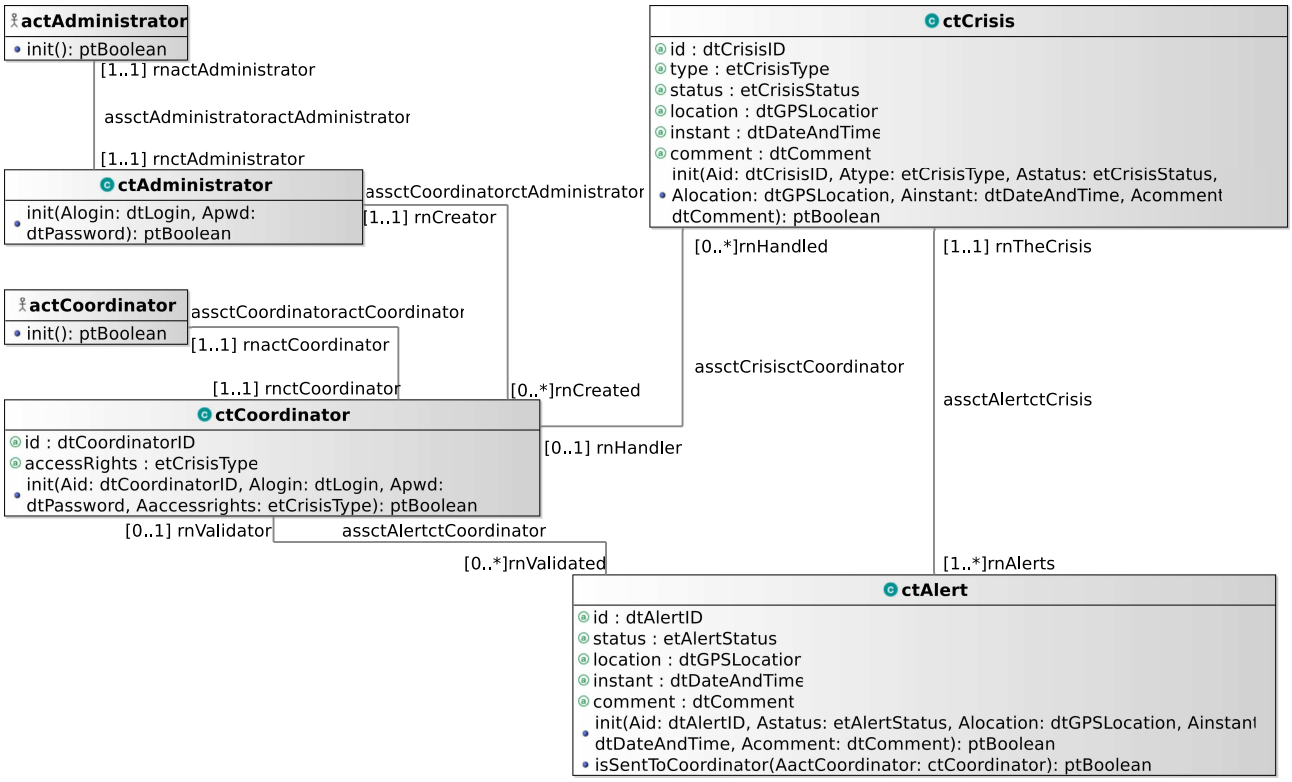

Figure 4.3: Concept Model - PrimaryTypes-Classes global view 06. ARvariant-CGV-01 concept global view 01.

4.2.2 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	
dtAdministratorID	
A string used to identify the administrator.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid administrator identifiers.
dtAlertID	
A string used to identify alerts.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid alert identifiers.
dtCaptcha	
A string used to store the Captcha information after the iCrash user failed to log in three times.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid dtCaptcha.
dtCity	
A string used to store the name of the city in which the point of interest is situated.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid dtCity.
dtComment	
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.
dtCoordinatorID	
A string used to identify coordinators.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid coordinators identifiers.
dtCrisisID	
A string used to identify crisis.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid crisis identifiers.
dtDescription	
A datatype made of a string value used to receive, store and send textual information about a point of interest.	
extends	dtString
operation	**is():ptBoolean**
	used to determine which strings are considered as valid descriptions.
dtGPSLocation	

continues in next page . . .

. . . Datatypes table continuation

Used to define coordinates of geographical positions on earth. It is defined a couple made of a latitude and a longitude.	
extends	dtString
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.
dtID	
A string used to identify points of interest and requests.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid points of interest and request identifiers.
dtIgnored	
A boolean used to store the value considering if the request should be ignored or not.	
operation	**is():ptBoolean** used to determine which boolean values are considered as valid dtIgnored.
dtLatitude	
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.
dtLogin	
A login string used to authentify an iCrash user.	
extends	dtString
operation	**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.	
extends	dtReal
operation	**is():ptBoolean** used to determine which strings are considered as valid dtLongitude.
dtMessage	
A string used to store the information that will be sent as input event to the actors.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid dtMessage.
dtName	
A string used to store the names of the points of interest.	
extends	dtString
operation	**is():ptBoolean** used to determine which strings are considered as valid names.
dtPassword	
A password string used to authentify an iCrash user.	
extends	dtString

continues in next page . . .

... Datatypes table continuation

operation	**is():ptBoolean**
	used to determine which strings are considered as valid dtPassword.
dtPhoneNumber	
A string used to store the phone number from the person declaring the crisis or the alert.	
extends	dtString
operation	**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.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.
etCategory	
This type is used to indicate the different categories of a point of interest.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.
etCrisisStatus	
This type is used to indicate the different handling status of a crisis.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.
etCrisisType	
This type is used to indicate the different types of a crisis.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.
etPersonType	
This type is used to indicate the type of person that informs about a car crash crisis.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.
etRequestStatus	
This type is used to indicate the different status of the request made for a point of interest.	
operation	**is():ptBoolean**
	used to determine which literal belongs to the enumeration.

4.2.3 Primary types - Association types descriptions

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

ASSOCIATIONS
assctAdministratoractAdministrator
frequent messages must be sent to coordinator especially in relation to coordinators they manage.
assctAlertctCoordinator
alerts need to be sent to coordinator in order to be validated or invalidated
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.

continues in next page ...

. . . Associations table continuation

assctAlertctPerson alerts are notified by person through the communication company. We need to keep an internal representation of those person 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.
assctCoordinatorctAdministrator one administrator manages more coordinators
assctCrisisctCoordinator at any point in time we need to know if a coordinator is handling existing crisis or not.
assctPersonactComCompany in order to communicate with person who informed about potential crisis, we need to record the communication company to use to send them messages.
assctPersonactPerson messages must be sent to person
assctPIctAdministrator one administrator is charge of handling various point of interest
assctPIctPerson several people can request several points of interests
assctRequestctAdministrator one administrator is charge of handling various point of interest requests
assctRequestctCoordinator several coordinators are charge of monitoring various point of interest requests

4.2.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

4.2.4.1 Primary types - Composition types descriptions

There are no composition types for the primary types.

4.2.5 Secondary types - Class types descriptions

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

4.2.6 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
dtSMS A datatype made of a string value used to send textual information to human mobile devices.
attribute **`value: ptString`** the textual information.

continues in next page . . .

. . . Datatypes table continuation

operation	**`is():ptBoolean`** used to determine which strings are considered as valid comments.

4.2.7 Secondary types - Association types descriptions

There are no association types for the secondary types.

4.2.8 Secondary types - Aggregation types descriptions

There are no aggregation types for the secondary types.

4.2.9 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 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 oeAddPI

The `oeAddPI` operation has the following properties:

Operation	
oeAddPI	
sent to add a new point of interest in the systems post state.	
Parameters	
1	**APIID: dtID** used to initialise the id field
2	**APIName: dtName** used to initialise the name field
3	**APICity: dtCity** used to initialise the city field
4	**APIGPSLocation: dtGPSLocation** used to initialise the gps location field
5	**APIDescription: dtDescription** used to initialise the description field
6	**APICategory: etCategory** used to initialise the category field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exist one ctRequest instance with the same id attribute than the one the administrator wants to create.

continues in next page . . .

... Operation table continuation

PreF 2	PreF02 it is supposed that there does NOT exist one ctPI instance with the same id attribute than the one the administrator wants to create.
Post-Condition (functional)	
PostF 1	PostF01 the new PI is added to the system.
PostF 2	PostF02 the administrator actor is informed about the satisfaction of its request.
PostF 3	PostF03 the person actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

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

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let TheActor:actAdministrator in
18     let ColctPI:Set(ctPI) in
19     let ColctRequest:Set(ctRequest) in
20
21     self.rnActor.rnSystem = TheSystem
22     and self.rnActor = TheActor
23
24     /* PreF01 */
25     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
26     and ColctPI->IsEmpty() = true
27
28     /* PreF02 */
29     and TheSystem.rnctRequest->select(id.eq(APIID) status.eq("solved")) = ColctRequest
30     and ColctRequest->IsEmpty() = false}
31
32 /* Post Functional:*/
33 postF{let TheSystem:ctState in
34     let TheActor:actAdministrator in
35     let ThectPI:ctPI in
36     let ThePerson:actPerson in
37
38     self.rnActor.rnSystem = TheSystem
39     and self.rnActor = TheActor
40
41     /* PostF01 */
42     ThectPI.init(APIID, APIName, APICity, APIGPSLocation, APIDescription, APICategory)
43
44     /* PostF02 */
45     and TheActor.rnInterfaceIN^iePIAdded()
46
47     /* PostF03 */
48     and ThePerson.rnInterfaceIN^iePIAdded()}
49
```

```
50 /* Post Protocol:*/
51 postP{ true}
```

Listing 5.1: Messir (MCL-oriented) specification of the operation *oeAddPI*.

5.1.2 Operation Model for oeUpdatePI

The `oeUpdatePI` operation has the following properties:

Operation	
oeUpdatePI	
sent to update an existing point of interest in the systems post state.	
Parameters	
1	**APIID: dtID** id used for ctPI instance retrieval
2	**APIName: dtName** name used for ctPI instance retrieval
3	**APICity: dtCity** city used for ctPI instance retrieval
4	**APIGPSLocation: dtGPSLocation** gps location used for ctPI instance retrieval
5	**APIDescription: dtDescription** description used for ctPI instance retrieval
6	**APICategory: etCategory** category used for ctPI instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctPI instance with the same id attribute than the one the administrator wants to delete.
Post-Condition (functional)	
PostF 1	PostF01 the PI is deleted from the system.
PostF 2	PostF02 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.2 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem:ctState in
4     let TheActor:actAdministrator in
5
6     self.rnActor.rnSystem = TheSystem
7     and self.rnActor = TheActor
8
```

```
9       /* PreP01 */
10      TheSystem.vpStarted = true
11
12      /* PreP02 */
13      and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15  /* Pre Functional:*/
16  preF{let TheSystem:ctState in
17      let TheActor:actAdministrator in
18      let ColctPI:Set(ctPI) in
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
25      and ColctPI->size().eq(1)}
26
27  /* Post Functional:*/
28  postF{let TheSystem:ctState in
29      let ThectPI:ctPI in
30      let TheActor:actAdministrator in
31
32      self.rnActor.rnSystem = TheSystem
33      and self.rnActor = TheActor
34
35      /* PostF01 */
36      ThectPI.rnctPI.name = APIName
37      and ThectPI.rnctPI.city = APIName
38      and ThectPI.rnctPI.location = AGPSLocation
39      and ThectPI.rnctPI.description = APIDescription
40      and ThectPI.rnctPI.category = APICategory
41
42      /* PostF02 */
43      and TheActor.rnInterfaceIN^iePIUpToDate()}
44
45  /* Post Protocol:*/
46  postP{ true}
```

Listing 5.2: Messir (MCL-oriented) specification of the operation *oeUpdatePI*.

5.1.3 Operation Model for oeDeletePI

The `oeDeletePI` operation has the following properties:

OPERATION	
oeDeletePI	
sent to delete an existing point of interest in the systems post state.	
Parameters	
1	**APIID: dtID** used for ctPI instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctPI instance with the same id attribute than the one the administrator wants to update.
Post-Condition (functional)	

continues in next page . . .

. . . Operation table continuation

PostF 1	PostF01 the PI attributes are updated in the system.
PostF 2	PostF02 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.3 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let TheActor:actAdministrator in
18     let ColctPI:Set(ctPI) in
19
20     self.rnActor.rnSystem = TheSystem
21     and self.rnActor = TheActor
22
23     /* PreF01 */
24     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
25     and ColctPI->size().eq(1)}
26
27 /* Post Functional:*/
28 postF{let TheSystem:ctState in
29     let TheActor:actAdministrator in
30     let ThectPI:ctPI in
31
32     self.rnActor.rnSystem = TheSystem
33     and self.rnActor = TheActor
34
35     /* PostF01 */
36     TheSystem.rnctPI->select(id.ed(APIID)) = ThectPI
37     and ThectPI.msrIsKilled
38
39     /* PostF02 */
40     and TheActor.rnInterfaceIN^iePIDeleted()}
41
42 /* Post Protocol:*/
43 postP{ true}
```

Listing 5.3: Messir (MCL-oriented) specification of the operation *oeDeletePI*.

5.1.4 Operation Model for oeGetAllRequestsFromCoordinator

The `oeGetAllRequestsFromCoordinator` operation has the following properties:

Operation

continues in next page . . .

... Operation table continuation

oeGetAllRequestsFromCoordinator	
sent to get all the requests checked by the actor ActCoordinator in the systems post state.	
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	PostF01 the actor ActAdministrator gets a list of requests to handle if there are pending requests.
PostF 2	PostF02 the administrator actor is also informed about the non-satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.4 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{true}
17
18 /* Post Functional:*/
19 postF{let TheSystem:ctState in
20     let TheActor:actAdministrator in
21     let ColctRequest:Set(ctRequest) in
22
23     self.rnActor.rnSystem = TheSystem
24     and self.rnActor = TheActor
25
26     TheSystem.rnctRequest->select(status.eq("pending")) = ColctRequest
27
28     and if (ColctRequest->IsEmpty() = false)
29     then (
30      /* PostF01 */
31      and TheActor.rnInterfaceIN^ieRequestList()
32     )
33     else (
34      /* PostF02 */
35      and AMessage.eq('No requested points of interest!')
36      and TheActor.rnInterfaceIN^ieMessage(AMessage)
37     )
38     endif}
39
```

```
40 /* Post Protocol:*/
41 postP{ true}
```

Listing 5.4: Messir (MCL-oriented) specification of the operation *oeGetAllRequestsFromCoordinator*.

5.1.5 Operation Model for oeTreatRequest

The `oeTreatRequest` operation has the following properties:

Operation	
oeTreatRequest	
sent to treat the requested PI in the systems post state.	
Parameters	
1	**ARequestID: dtID**
	id used for ctRequest instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same id attribute and the status pending than the one the administrator wants to treat.
Post-Condition (functional)	
PostF 1	PostF01 the requests status is updated to treated in the system.
PostF 2	PostF02 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.5 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let TheActor:actAdministrator in
18     let ColctRequest:Set(ctRequest) in
19
20     self.rnActor.rnSystem = TheSystem
21     and self.rnActor = TheActor
```

```
22
23      /* PreF01 */
24      TheSystem.rnctRequest->select(status.eq("pending")) = ColctRequest
25      and ColctRequest->IsEmpty() = false}
26
27  /* Post Functional:*/
28  postF{let TheSystem:ctState in
29      let TheActor:actAdministrator in
30      let ThectRequest:ctRequest in
31
32      self.rnActor.rnSystem = TheSystem
33      and self.rnActor = TheActor
34
35      /* PostF01 */
36      ThectRequest.rnctRequest.status = "treated"
37
38      /* PostF02 */
39      and TheActor.rnInterfaceIN^ieRequestBeingTreated()}
40
41  /* Post Protocol:*/
42  postP{ true}
```

Listing 5.5: Messir (MCL-oriented) specification of the operation *oeTreatRequest*.

5.1.6 Operation Model for oeSolveRequest

The `oeSolveRequest` operation has the following properties:

Operation	
oeSolveRequest	
sent to solve the requested PI in the systems post state.	
Parameters	
1	**ARequestID: dtID** id used for ctRequest instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same id attribute and the status treated than the one the administrator wants to solve.
Post-Condition (functional)	
PostF 1	PostF01 the request's status is updated to solved in the system.
PostF 2	PostF02 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.6 provides the Messir (MCL-oriented) specification of the operation.

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

```
4       let TheActor:actAdministrator in
5
6       self.rnActor.rnSystem = TheSystem
7       and self.rnActor = TheActor
8
9       /* PreP01 */
10      TheSystem.vpStarted = true
11
12      /* PreP02 */
13      and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15  /* Pre Functional:*/
16  preF{let TheSystem:ctState in
17      let TheActor:actAdministrator in
18      let ColctRequest:Set(ctRequest) in
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctRequest->select(status.eq("treated")) = ColctRequest
25      and ColctRequest->IsEmpty() = false}
26
27  /* Post Functional:*/
28  postF{let TheSystem:ctState in
29      let TheActor:actAdministrator in
30      let ThectRequest:ctRequest in
31
32      self.rnActor.rnSystem = TheSystem
33      and self.rnActor = TheActor
34
35      /* PostF01 */
36      ThectRequest.rnctRequest.status = "solved"
37
38      /* PostF02 */
39      and TheActor.rnInterfaceIN^ieRequestSolved()}
40
41  /* Post Protocol:*/
42  postP{ true}
```

Listing 5.6: Messir (MCL-oriented) specification of the operation *oeSolveRequest*.

5.1.7 Operation Model for oeAddCoordinator

The `oeAddCoordinator` operation has the following properties:

OPERATION	
oeAddCoordinator	
sent to add a new coordinator in the systems post state and environments post state.	
Parameters	
1	**AdtCoordinatorID: dtCoordinatorID** used to initialize the id field
2	**AdtLogin: dtLogin** used to initialize the login field
3	**AdtPassword: dtPassword** used to initialize the password field
4	**CoordinatorAccessRights: etCrisisType** used to initialize the access right field
Return type	
ptBoolean	

continues in next page . . .

... Operation table continuation

Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there cannot exist a ctCoordinator instance with the same id attribute as the one the administrator wants to delete.
Post-Condition (functional)	
PostF 1	PostF 1 the environment has a new instance of coordinator actor allowing for input/output message communication with the system.
PostF 2	PostF 2 the systems state has a new instance of ctCoordinator initialized with the given values.
PostF 3	PostF 3 the new actor instance and ctCoordinator instance are related.
PostF 4	PostF 4 the new actor instance and ctCoordinator instance are related according to the authenticated association.
PostF 5	PostF 5 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.7 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem: ctState in
17     let TheActor:actAdministrator in
18     let ColctCoordinators:Bag(ctCoordinator) in
19
20     self.rnActor.rnSystem = TheSystem
21     and self.rnActor = TheActor
22
23     /* PreF01 */
24     and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
25     and ColctCoordinators->isEmpty() = true}
26
27 /* Post Functional:*/
28 postF{let TheSystem: ctState in
29     let TheactCoordinator:actCoordinator in
30     let ThectCoordinator:ctCoordinator in
31
32     self.rnActor.rnSystem = TheSystem
33     and self.rnActor = TheActor
34
35     /* PostF01 */
```

```
36      TheactCoordinator.init()
37
38      /* PostF02 */
39      and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword, CoordinatorAccessRights)
40
41      /* PostF03 */
42      and TheactCoordinator@post.rnctCoordinator = ThectCoordinator
43
44      /* PostF04 */
45      and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator
46
47      /* PostF05 */
48      and TheActor.rnInterfaceIN^ieCoordinatorAdded()}
49
50  /* Post Protocol:*/
51  postP{ true}
```

Listing 5.7: Messir (MCL-oriented) specification of the operation *oeAddCoordinator*.

5.1.8 Operation Model for oeDeleteCoordinator

The `oeDeleteCoordinator` operation has the following properties:

OPERATION	
oeDeleteCoordinator sent to delete an existing coordinator in the systems post state and environments post state.	
Parameters	
1	**AdtCoordinatorID: dtCoordinatorID** used for ctCoordinator instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started.
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one ctCoordinator instance with the same id attribute than the one the administrator wants to delete.
Post-Condition (functional)	
PostF 1	PostF 1 the ctCoordinator class instance having the required id do not belong anymore to the post state as well as its related actCoordinator actor instance.
PostF 2	PostF 2 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.8 provides the Messir (MCL-oriented) specification of the operation.

```
1
2   /* Pre Protocol:*/
3   preP{let TheSystem:ctState in
4       let TheActor:actAdministrator in
5
```

```
6       self.rnActor.rnSystem = TheSystem
7       and self.rnActor = The Actor
8
9       /* Pre P01 */
10      TheSystem.vpStarted = true
11
12      /* Pre P02 */
13      TheActor.rnctAuthenticated.vpIsLogged = true}
14
15  /* Pre Functional:*/
16  preF{let TheSystem: ctState in
17      let TheActor:actAdministrator in
18      let ColctCoordinators:Bag(ctCoordinator) in
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
25      and ColctCoordinators->size().eq(1)}
26
27  /* Post Functional:*/
28  postF{let TheSystem: ctState in
29      let TheActor:actAdministrator in
30      let ThectCoordinator:ctCoordinator in
31
32      self.rnActor.rnSystem = TheSystem
33      and self.rnActor = TheActor
34
35      /* PostF01 */
36      TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ThectCoordinator
37      and ThectCoordinator.rnactCoordinator->forAll(msrIsKilled)
38      and ThectCoordinator.msrIsKilled
39
40      /* PostF02 */
41      and TheActor.rnInterfaceIN^ieCoordinatorDeleted()
42
43      /* Post Protocol:*/
44       * /* PostP01 */
45      and true}
46
47  /* Post Protocol:*/
48  postP{ true}
```

Listing 5.8: Messir (MCL-oriented) specification of the operation *oeDeleteCoordinator*.

5.1.9 Operation Model for oeUpdateCoordinatorAccessRights

The `oeUpdateCoordinatorAccessRights` operation has the following properties:

OPERATION	
oeUpdateCoordinatorAccessRights sent to update an existing coordinators access rights in the systems post state and environments post state.	
Parameters	
1	**AdtCoordinatorID: dtCoordinatorID** used for ctCoordinator instance retrieval
2	**CoordinatorAccessRights: etCrisisType** used to update access rights field
Return type	
ptBoolean	
Pre-Condition (protocol)	

continues in next page . . .

. . . Operation table continuation

PreP 1	PreP 1 the system is started PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
Pre-Condition (functional)	
PreF 1	it is supposed that there exist one ctCoordinator instance with the same id attribute than the one the administrator wants to update.
Post-Condition (functional)	
PostF 1	PostF 1 the ctCoordinator class instance having the required id has its access rights updated in the post state as well as its related actCoordinator actor instance. PostF 2 the administrator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.9 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actAdministrator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem: ctState in
17     let TheActor:actAdministrator in
18     let ColctCoordinators:Bag(ctCoordinator) in
19
20     self.rnActor.rnSystem = TheSystem
21     and self.rnActor = TheActor
22
23     /* PreF01 */
24     TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
25     and ColctCoordinators->size().eq(1)}
26
27 /* Post Functional:*/
28 postF{let TheSystem: ctState in
29     let TheActor:actAdministrator in
30     let ThectCoordinator:ctCoordinator in
31
32     self.rnActor.rnSystem = TheSystem
33     and self.rnActor = TheActor
34
35     /* PostF01 */
36     TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ThectCoordinators
37     and ThectCoordinator.accessRights.eq(AaccessRights)
38
39     /* PostF02 */
40     and TheActor.rnInterfaceIN^ieCoordinatorAccessRightsUpdated()}
41
42 /* Post Protocol:*/
43 postP{ true}
```

Listing 5.9: Messir (MCL-oriented) specification of the operation

oeUpdateCoordinatorAccessRights.

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 sent to request authorization to request access secured system operations.	
Parameters	
1	**ALogin: dtLogin** first information used to determine accessibility rights for the actual actor.
2	**APassword: dtPassword** second information used to determine accessibility rights for the actual actor.
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor is not already logged in ! (i.e. the associated ctAuthenticated instance is not considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 none
Post-Condition (functional)	
PostF 1	PostF 1 if the login and password provided by the actor correspond to the ones that belong to the ctAuthenticated instance he is related to then a welcome message is sent to the actor (n.b. the logged status is changed as a post-protocol condition); else the actor is notified that he gave incorrect data and all the administrator actors existing in the environment are notified of an intrusion attempt.
Post-Condition (protocol)	
PostP 1	PostP 1 if the authentication information is correct then the actor is known to be logged in ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged)

The listing 5.10 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4     let TheActor:actAuthenticated in
5     self.rnActor.rnSystem = TheSystem
6     and self.rnActor = TheActor
7
8     /* PreP01 */
9     and TheSystem.vpStarted = true
10    /* PreP02 */
```

```
11     and TheActor.rnctAuthenticated.vpIsLogged = false
12     /* PreP03 */
13     and TheActor.rnctAuthenticated.capReq = false}
14
15  /* Pre Functional:*/
16  preF{/* PreF01 */
17     true}
18
19  /* Post Functional:*/
20  postF{let TheSystem: ctState in
21     let TheactAuthenticated:actAuthenticated in
22
23     let AptStringMessageForTheactAuthenticated: ptString in
24     let AptStringMessageForTheactAdministrator:ptString in
25
26     self.rnActor.rnSystem = TheSystem
27     and self.rnActor = TheactAuthenticated
28
29     and /* PostF01 */
30      if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
31       and TheactAuthenticated.rnctAuthenticated.login = AdtLogin )
32      then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
33       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
34       )
35      else (AptStringMessageForTheactAuthenticated
36       .eq('Wrong identification information ! Please try again ...')
37       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
38       and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
39       and TheSystem.rnactAdministrator
40       .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
41       )
42      endif}
43
44  /* Post Protocol:*/
45  postP{ let TheSystem: ctState in
46     let TheactAuthenticated:actAuthenticated in
47
48     self.rnActor.rnSystem = TheSystem
49     and self.rnActor = TheactAuthenticated
50     /* PostP01 */
51     if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword and
52     TheactAuthenticated.rnctAuthenticated.login = AdtLogin) then(
53      TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true and
54      TheactAuthenticated.rnctAuthenticated@post.tries = 0 and
55      TheactAuthenticated.rnctAuthenticated@post.lastAccess = -181
56     )else(
57      TheactAuthenticated.rnctAuthenticated@post.lastAccess = ((int)System.currentTimeMillis()/1000)
            - TheactAuthenticated.rnctAuthenticated.lastAccess
58      if(TheactAuthenticated.rnctAuthenticated@post.lastAccess <= 180)then(
59       TheactAuthenticated.rnctAuthenticated@post.tries = TheactAuthenticated.rnctAuthenticated@post.
             tries + 1
60       if (TheactAuthenticated.rnctAuthenticated@post.tries > 2) then(
61        TheactAuthenticated.rnctAuthenticated@post.capReq = true
62       )else
63        true
64       endif
65      )else(
66       TheactAuthenticated.rnctAuthenticated@post.tries = 1
67      )endif
68     )endif}
```

Listing 5.10: Messir (MCL-oriented) specification of the operation *oeLogin*.

5.2.2 Operation Model for oeLogout

The `oeLogout` operation has the following properties:

OPERATION	
oeLogout	
sent to end the secured access to specific system operations.	
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor is currently logged in ! (i.e. the associated ctAuthenticated instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 none
Post-Condition (functional)	
PostF 1	PostF 1 a logout confirmation message is sent to the actor (n.b. the logged status is changed as a post-protocol condition)
Post-Condition (protocol)	
PostP 1	PostP 1 the actor is known to be logged out ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged out)

The listing 5.11 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4     let TheActor:actAdministrator in
5     self.rnActor.rnSystem = TheSystem
6     and self.rnActor = TheActor
7
8     /* PreP01 */
9     and TheSystem.vpStarted = true
10     /* PreP02 */
11    and TheActor.rnctAuthenticated.vpIsLogged = true}
12
13 /* Pre Functional:*/
14 preF{/* PreF01 */
15    true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19    let TheactAuthenticated:actAuthenticated in
20    let AptStringMessageForTheactAuthenticated: ptString in
21
22    self.rnActor.rnSystem = TheSystem
23    and self.rnActor = TheactAuthenticated
24
25    /* PostF01 */
26    AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
27    and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)}
28
29 /* Post Protocol:*/
30 postP{ let TheSystem: ctState in
31    let TheactAuthenticated:actAuthenticated in
32
33    self.rnActor.rnSystem = TheSystem
34    and self.rnActor = TheactAuthenticated.asSet
35    /* PostP01 */
```

```
36    TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false}
```

Listing 5.11: Messir (MCL-oriented) specification of the operation *oeLogout*.

5.2.3 Operation Model for oeLoginWithCaptcha

The `oeLoginWithCaptcha` operation has the following properties:

Operation	
oeLoginWithCaptcha	
This operation can and must only be done when capReq boolean value is true. Used to authenticate the user and to also authenticate the the user is not a robot.	
Parameters	
1	**ALogin: dtLogin**
	The username of the actor.
2	**APassword: dtPassword**
	The key to login of the actor.
3	**ACaptcha: dtCaptcha**
	The captcha the actor sent to the system to compare to the captcha to solve.
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	
Pre-Condition (functional)	
PreF 1	
Post-Condition (functional)	
PostF 1	
Post-Condition (protocol)	
PostP 1	

The listing 5.12 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4     let TheActor:actAuthenticated in
5     self.rnActor.rnSystem = TheSystem
6     and self.rnActor = TheActor
7
8     /* PreP01 */
9     and TheSystem.vpStarted = true
10    /* PreP02 */
11    and TheActor.rnctAuthenticated.vpIsLogged = false
12    /* PreP03 */
13    and TheActor.rnctAuthenticated.capReq = true}
14
15 /* Pre Functional:*/
16 preF{/* PreF01 */
17    true}
18
19 /* Post Functional:*/
20 postF{let TheSystem: ctState in
21    let TheactAuthenticated:actAuthenticated in
```

```
22
23      let AptStringMessageForTheactAuthenticated:ptString in
24      let AptStringMessageForTheactAdministrator:ptString in
25
26      self.rnActor.rnSystem = TheSystem
27      and self.rnActor = TheactAuthenticated
28
29      and /* PostF01 */
30      if(ACaptcha = TheactAuthenticated.rnctAuthenticated.cap2Solve) then
31       if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
32        and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
33        )
34       then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
35        and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
36        )
37       else (AptStringMessageForTheactAuthenticated
38        .eq('Wrong identification information ! Please try again ...')
39        and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
40        and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
41        and TheSystem.rnactAdministrator
42        .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
43        )
44       endif
45      else
46       (TheactAuthenticated.rnctAuthenticated@post.cap2Solve = random() and
47        AptStringMessageForTheactAuthenticated.eq('Wrong Captcha information ! Captcha: ' +
48         TheactAuthenticated.rnctAuthenticated@post.cap2Solve) and
49        TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
50       )
51      endif}
52
53  /* Post Protocol:*/
54  postP{ let TheSystem: ctState in
55      let TheactAuthenticated:actAuthenticated in
56
57      self.rnActor.rnSystem = TheSystem
58      and self.rnActor = TheactAuthenticated
59      /* PostP01 */
60      if(ACaptcha = TheactAuthenticated.rnctAuthenticated.cap2Solve) then
61       (if(TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword and
62       TheactAuthenticated.rnctAuthenticated.login = AdtLogin) then(
63        TheactAuthenticated.rnctAuthenticated@post.tries = 0 and
64        TheactAuthenticated.rnctAuthenticated@post.lastAccess = -181 and
65        TheactAuthenticated.rnctAuthenticated@post.capReq = false and
66        TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true
67       )else(
68        TheactAuthenticated.rnctAuthenticated@post.lastAccess = ((int)System.currentTimeMillis()/1000)
              - TheactAuthenticated.rnctAuthenticated.lastAccess
69        (if(TheactAuthenticated.rnctAuthenticated@post.lastAccess <= 180)then(
70         TheactAuthenticated.rnctAuthenticated@post.tries = TheactAuthenticated.rnctAuthenticated.
               tries + 1
71        )else(
72         TheactAuthenticated.rnctAuthenticated@post.tries = 1
73        )endif)
74       )endif)
75      else
76       (false)
77      endif}
```

Listing 5.12: Messir (MCL-oriented) specification of the operation *oeLoginWithCaptcha*.

5.2.4 Operation Model for oeResetPassword

The `oeResetPassword` operation has the following properties:

OPERATION

continues in next page . . .

... Operation table continuation

oeResetPassword
Operation used to reset a users password, does not reset the password of an admin tho.
Parameters
1 **ALogin: dtLogin** The username of the actor to reset the password.
Return type
ptBoolean
Pre-Condition (protocol)
PreP 1
Pre-Condition (functional)
PreF 1
Post-Condition (functional)
PostF 1
Post-Condition (protocol)
PostP 1

The listing 5.13 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4     let TheActor:actAuthenticated in
5     self.rnActor.rnSystem = TheSystem
6     and self.rnActor = TheActor
7
8     /* PreP01 */
9     and TheSystem.vpStarted = true
10    /* PreP02 */
11    and TheActor.rnctAuthenticated.vpIsLogged = false}
12
13 /* Pre Functional:*/
14 preF{/* PreF01 */
15    true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19    let TheactAuthenticated:actAuthenticated in
20
21    let AptStringMessageForTheactAuthenticated: ptString in
22    let AptStringMessageForTheactAdministrator:ptString in
23
24    self.rnActor.rnSystem = TheSystem
25    and self.rnActor = TheactAuthenticated
26
27    and /* PostF01 */
28     if (TheactAuthenticated.rnctAuthenticated.login = AdtLogin
29      )
30     then (TheactAuthenticated.rnctAuthenticated@post.pwd = new dtPassword( random() )
31      and AptStringMessageForTheactAuthenticated.eq('User found, resetting password ... New password
            = ' + TheactAuthenticated.rnctAuthenticated@post.pwd)
32      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
33      )
34     else (AptStringMessageForTheactAuthenticated
35      .eq('Wrong username information ! Please try again ...')
36      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
37      )
38     endif}
```

```
39
40 /* Post Protocol:*/
41 postP{ }
```

Listing 5.13: Messir (MCL-oriented) specification of the operation *oeResetPassword*.

5.3 Environment - Out Interface Operation Scheme for actCoordinator

5.3.1 Operation Model for oeGetAllRequests

The `oeGetAllRequests` operation has the following properties:

Operation	
oeGetAllRequests	
sent to get a list of requested PIs.	
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	PostF01 the actor ActCoordinator gets a list of requests to handle if there are unhandled requests (no status).
PostF 2	PostF02 the coordinator actor is also informed about the non-satisfaction of its request.
Post-Condition (protocol)	
PostP 1	non

The listing 5.14 provides the Messir (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem:ctState in
4     let TheActor:actAdministrator in
5
6     self.rnActor.rnSystem = TheSystem
7     and self.rnActor = TheActor
8
9     /* PreP01 */
10    TheSystem.vpStarted = true
11
12    /* PreP02 */
13    and TheActor.rnctAuthenticated.vpIsLogged = true
14
15    /* Pre P03 */
16    if(ctCrisis.type = huge)
17    then(
18     TheActor.accessRights = huge
19    )
20    else(
21     if(ctCrisis.type = medium)
```

```
22      then(
23       TheActor.accessRights = huge
24       or TheActor.accessRights = medium
25      )
26      else(
27       TheActor.accessRights = huge
28       or TheActor.accessRights = medium
29       or TheActor.accessRights = small
30      )
31      endif
32     )
33     endif}
34
35 /* Pre Functional:*/
36 preF{true}
37
38 /* Post Functional:*/
39 postF{let TheSystem:ctState in
40     let TheActor:actCoordinator in
41     let ColctRequest:Set(ctRequest) in
42
43     self.rnActor.rnSystem = TheSystem
44     and self.rnActor = TheActor
45
46     TheSystem.rnctRequest->select(status.eq("")) = ColctRequest
47
48     and if (ColctRequest->IsEmpty() = false)
49     then (
50      /* PostF01 */
51      and TheActor.rnInterfaceIN^ieRequestListToCheck()
52     )
53     else (
54      /* PostF02 */
55      and AMessage.eq('No requests to check!')
56      and TheActor.rnInterfaceIN^ieMessage(AMessage)
57     )
58     endif}
59
60 /* Post Protocol:*/
61 postP{ true}
```

Listing 5.14: Messip (MCL-oriented) specification of the operation *oeGetAllRequests*.

5.3.2 Operation Model for oeCheckAvailability

The `oeCheckAvailability` operation has the following properties:

OPERATION	
oeCheckAvailability	
sent to check if the requested PI is in the system.	
Parameters	
1	**ARequestID: dtID** id used for ctRequest instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same name, city and category attributes than the one the coordinator wants to compare to.

continues in next page . . .

. . . Operation table continuation

Post-Condition (functional)	
PostF 1	PostF01 the request's ignored is updated to 'false' in the system if the requested PI is NOT in the system.
PostF 2	PostF02 the request's ignored is updated to 'true' in the system if the requested PI is in the system.
PostF 3	PostF03 the coordinator actor is informed about the satisfaction or non-satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.15 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4       let TheActor:actAdministrator in
5
6       self.rnActor.rnSystem = TheSystem
7       and self.rnActor = TheActor
8
9       /* PreP01 */
10      TheSystem.vpStarted = true
11
12      /* PreP02 */
13      and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17      let TheActor:actCoordinator in
18      let ColctPI:Set(ctPI) in
19      let ColctRequest:Set(ctRequest) in
20
21      self.rnActor.rnSystem = TheSystem
22      and self.rnActor = TheActor
23
24      /* PreF01 */
25      TheSystem.rnctRequest->select(name.eq(ARequestID)) = ColctRequest
26      and ColctRequest->size().eq(1)}
27
28 /* Post Functional:*/
29 postF{let TheSystem:ctState in
30      let TheActor:actCoordinator in
31      let ThectRequest:ctRequest in
32
33      self.rnActor.rnSystem = TheSystem
34      and self.rnActor = TheActor
35
36      TheSystem.rnctPI->select(name.eq(ARequestID)) = ColctPI
37
38      and if (ColctPI.IsEmpty() = false)
39      then (
40       /* PostF01 */
41       and ThectRequest.rnctRequest.ignored = true
42       and AMessage.eq('The PI is already in the system.')
43       and TheActor.rnInterfaceIN^ieMessage(AMessage)
44      )
45      else (
46       /* PostF02 */
47       and ThectRequest.rnctRequest.ignored = false
48       and AMessage.eq('The PI is NOT in the system. How to proceed?')
```

```
49      and TheActor.rnInterfaceIN^ieMessage(AMessage)
50     )
51     endif}
52
53 /* Post Protocol:*/
54 postP{ true}
```

Listing 5.15: Messir (MCL-oriented) specification of the operation *oeCheckAvailability*.

5.3.3 Operation Model for oeDeliverRequest

The `oeDeliverRequest` operation has the following properties:

OPERATION	
oeDeliverRequest	
sent to change the status of a requested PI in the systems post state.	
Parameters	
1	**ARequestID: dtID** id used for ctRequest instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same id attribute than the one the coordinator wants to change the status.
Post-Condition (functional)	
PostF 1	PostF01 the request's status is updated to 'pending' in the system's post state.
PostF 2	PostF02 the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.16 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let TheActor:actCoordinator in
18     let ColctRequest:Set(ctRequest) in
```

```
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctRequest->select(ignored.eq(false)) = ColctRequest
25      and ColctRequest->IsEmpty() = false}
26
27  /* Post Functional:*/
28  postF{let TheSystem:ctState in
29      let TheActor:actCoordinator in
30      let ThectRequest:ctRequest in
31
32      self.rnActor.rnSystem = TheSystem
33      and self.rnActor = TheActor
34
35      /* PostF01 */
36      ThectRequest.rnctRequest.status = "pending"
37
38      /* PostF02 */
39      and TheActor.rnInterfaceIN^ieRequestDelivered()}
40
41  /* Post Protocol:*/
42  postP{ true}
```

Listing 5.16: Messir (MCL-oriented) specification of the operation *oeDeliverRequest*.

5.3.4 Operation Model for oeValidateAlert

The `oeValidateAlert` operation has the following properties:

Operation	
oeValidateAlert sent to indicate that a specific alert is not a fake.	
Parameters	
1	**AdtAlertID: dtAlertID** the identification information used to determine the alert instance
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one ctAlert instance with the same id attribute value as the one provided by the coordinator actor who wants to validate.
Post-Condition (functional)	
PostF 1	PostF 1 the ctAlert class instance having the provided id is considered as valid in the post state and the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.17 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let ColctAlert:Bag(ctAlert) in
18
19     self.rnActor.rnSystem = TheSystem
20
21     /* Pre F01 */
22     TheSystem.rnctAlert->select(id.eq(AdtAlertID)) = ColctAlert
23     and ColctAlert->size.eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem:ctState in
27     let ThectAlert:ctAlert in
28
29     self.rnActor.rnSystem = TheSystem
30
31     /* Post F01 */
32     TheSystem.rnctAlert->select(id.eq(AdtAlerID)) = ThectAlert
33     and ThectAlert.status.eq("valid")
34     and TheActor.rnInterfaceIN^ieMessage("Alert validated!")}
35
36 /* Post Protocol:*/
37 postP{ true}
```

Listing 5.17: Messir (MCL-oriented) specification of the operation *oeValidateAlert*.

5.3.5 Operation Model for oeInvalidateAlert

The `oeInvalidateAlert` operation has the following properties:

OPERATION	
oeInvalidateAlert	
sent to indicate that an alert should be considered as closed.	
Parameters	
1	**AdtAlertID: dtAlertID** the identification information used to determine the alert to close
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one ctAlert instance with the same id attribute value as the one provided by the coordinator actor who wants to close.

continues in next page . . .

... Operation table continuation

Post-Condition (functional)	
PostF 1	PostF 1 the ctAlert class instance having the provided id is considered closed in the post state.
PostF 2	PostF 2 the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.18 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let ColctAlert:Bag(ctAlert) in
18
19     self.rnActor.rnSystem = TheSystem
20
21     /* Pre F01 */
22     TheSystem.rnctAlert->select(id.eq(AdtAlertID)) = ColctAlert
23     and ColctAlert->size.eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem:ctState in
27     let ThectAlert:ctAlert in
28
29     self.rnActor.rnSystem = TheSystem
30
31     /* Post F01 */
32     TheSystem.rnctAlert->select(id.eq(AdtAlerID)) = ThectAlert
33     and ThectAlert.status.eq("closed")
34     and TheActor.rnInterfaceIN^ieMessage("Alert invalidated!")}
35
36 /* Post Protocol:*/
37 postP{ true}
```

Listing 5.18: Messir (MCL-oriented) specification of the operation *oeInvalidateAlert*.

5.3.6 Operation Model for oeGetAlertSet

The `oeGetAlertSet` operation has the following properties:

OPERATION
oeGetAlertSet
sent to request all the ctAlert instances having a specific status.

continues in next page ...

. . . Operation table continuation

Parameters	
1	**AetAlertStatus: etAlertStatus** the criteria used to select the alerts to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 none
Post-Condition (functional)	
PostF 1	PostF 1 the post state is the one obtained by satisfying the isSentToCoordinator predicate for each alert having the provided status and for the actor sending the message. (cf. specification of isSentToCoordinator predicate given for the ctAlert type.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.19 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{true}
17
18 /* Post Functional:*/
19 postF{let TheSystem: ctState in
20     let ThectAlert:ctAlert in
21
22     self.rnActor.rnSystem = TheSystem
23
24     /* Post F01 */
25     TheSystem.rnctAlert->select(status.eq(AetAlertStatus)) = ThectAlert
26     and ThectAlert.rnctAlert->forAll(isSentToCoordinator)
27     and ThectAlert.msrIsSentToCoordinator}
28
29 /* Post Protocol:*/
30 postP{ true}
```

Listing 5.19: Messir (MCL-oriented) specification of the operation *oeGetAlertSet*.

5.3.7 Operation Model for oeGetCrisisSet

The `oeGetCrisisSet` operation has the following properties:

OPERATION	
oeGetCrisisSet	
sent to request all the ctCrisis instances having a specific status.	
Parameters	
1	**AetCrisisStatus: etCrisisStatus** the status information used to determine the crisis to send back to the actor
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 none
Post-Condition (functional)	
PostF 1	PostF 1 the post state is the one obtained by satisfying the isSentToCoordinator predicate for each crisis having the provided status and for the actor sending the message ieSendACrisis. (cf. specification of isSentToCoordinator predicate given for the ctCrisis type.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.20 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{true}
17
18 /* Post Functional:*/
19 postF{let TheSystem: ctState in
20     let ThectCrisis:ctCrisis in
21
22     self.rnActor.rnSystem = TheSystem
23
24     /* Post F01 */
25     TheSystem.rnctCrisis->select(status.eq(AetCrisisStatus)) = ThectCrisis
26     and ThectCrisis.rnctCrisis->forAll(isSentToCoordinator)
27     and ThectCrisis.msrIsSentToCoordinator}
28
```

```
29 /* Post Protocol:*/
30 postP{ true}
```

Listing 5.20: Messir (MCL-oriented) specification of the operation *oeGetCrisisSet*.

5.3.8 Operation Model for oeSetCrisisType

The `oeSetCrisisType` operation has the following properties:

Operation	
oeSetCrisisType	
sent to define the gravity type of a specific crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis
2	**AetCrisisType: etCrisisType** the new type value
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	PostF 1 the crisis type attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.21 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let ColctCrisis:Bag(ctCrisis) in
18
19     self.rnActor.rnSystem = TheSystem
```

```
20
21      /* Pre F01 */
22      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
23      and ColctCrisis->size.eq(1)}
24
25  /* Post Functional:*/
26  postF{let TheSystem:ctState in
27      let ThectCrisis:ctCrisis in
28
29      self.rnActor.rnSystem = TheSystem
30
31      /* Post F01 */
32      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
33      and ThectCrisis.AetCrisisType.eq(AdtCrisisType)
34      and TheActor.rnInterfaceIN^ieMessage("Crisis type set to " + AdtCrisisType)}
35
36  /* Post Protocol:*/
37  postP{ true}
```

Listing 5.21: Messir (MCL-oriented) specification of the operation *oeSetCrisisType*.

5.3.9 Operation Model for oeSetCrisisStatus

The `oeSetCrisisStatus` operation has the following properties:

OPERATION	
oeSetCrisisStatus	
sent to define the handling status of a specific crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis
2	**AetCrisisStatus: etCrisisStatus** the new status value
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	PostF 1 the crisis status attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.22 provides the Messir (MCL-oriented) specification of the operation.

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

```
4       let TheActor:actCoordinator in
5
6       self.rnActor.rnSystem = TheSystem
7       and self.rnActor = The Actor
8
9       /* Pre P01 */
10      TheSystem.vpStarted = true
11
12      /* Pre P02 */
13      TheActor.rnctAuthenticated.vpIsLogged = true}
14
15  /* Pre Functional:*/
16  preF{let TheSystem:ctState in
17      let ColctCrisis:Bag(ctCrisis) in
18
19      self.rnActor.rnSystem = TheSystem
20
21      /* Pre F01 */
22      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
23      and ColctCrisis->size.eq(1)}
24
25  /* Post Functional:*/
26  postF{let TheSystem:ctState in
27      let ThectCrisis:ctCrisis in
28
29      self.rnActor.rnSystem = TheSystem
30
31      /* Post F01 */
32      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
33      and ThectCrisis.AetCrisisStatus.eq(AdtCrisisStatus)
34      and TheActor.rnInterfaceIN^ieMessage("Crisis status set to " + AdtCrisisStatus)}
35
36  /* Post Protocol:*/
37  postP{ true}
```

Listing 5.22: Messir (MCL-oriented) specification of the operation *oeSetCrisisStatus*.

5.3.10 Operation Model for oeSetCrisisHandler

The `oeSetCrisisHandler` operation has the following properties:

OPERATION	
oeSetCrisisHandler	
sent to declare himself as been the handler of a crisis having the specified id.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 there exist one crisis having the given id in the pre-state.
Post-Condition (functional)	
PostF 1	PostF 1 the ctCrisis instance having the provided id is in handled status at poststate and is associated to the actor that sends the message (which himself is notified with a textual message as confirmation).

continues in next page . . .

... Operation table continuation

PostF 2	PostF 2 All the alerts related to this crisis are sent to the actor such that he can decide how to handle them.
PostF 3	PostF 3 if the crisis was already handled at pre-sate then the associated handler actor is notified about the change of handler for one of his crisis (n.b. it might be the same even if not relevant).
PostF 4	PostF 4 a message is sent to the communication company for any human related to an alert associated to the crisis. A human will receive as many messages as alerts he sent despite the fact that they might relate to the same crisis (i.e. one alert, one acknoledgement).
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.23 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let ColctCrisis:Bag(ctCrisis) in
18
19     self.rnActor.rnSystem = TheSystem
20
21     /* Pre F01 */
22     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
23     and ColctCrisis->size.eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem:ctState in
27     let ThectCrisis:ctCrisis in
28
29     self.rnActor.rnSystem = TheSystem
30
31     /* Post F01 */
32     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
33     and ThectCrisis.AetCrisisStatus.eq("handled")
34     and TheActor.rnInterfaceIN^ieMessage("Handler of the crisis " + AdtCrisisID + " is being
           handled by coordinator with id " + self.id)}
35
36 /* Post Protocol:*/
37 postP{ true}
```

Listing 5.23: Messir (MCL-oriented) specification of the operation *oeSetCrisisHandler*.

5.3.11 Operation Model for oeReportOnCrisis

The `oeReportOnCrisis` operation has the following properties:

OPERATION	
oeReportOnCrisis	
sent to update the textual information available for a specific handled crisis.	
Parameters	
1	**AdtCrisisID: dtCrisisID**
	the identification information used to determine the crisis to report on
2	**AdtComment: dtComment**
	the textual information commenting the crisis
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one crisis in the pre state having the given id.
Post-Condition (functional)	
PostF 1	PostF 1 the comment attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.24 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = The Actor
8
9      /* Pre P01 */
10     TheSystem.vpStarted = true
11
12     /* Pre P02 */
13     TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let ColctCrisis:Bag(ctCrisis) in
18
19     self.rnActor.rnSystem = TheSystem
20
21     /* Pre F01 */
22     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
23     and ColctCrisis->size.eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem:ctState in
27     let ThectCrisis:ctCrisis in
28
29     self.rnActor.rnSystem = TheSystem
30
31     /* Post F01 */
```

```
32      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
33      and ThectCrisis.AdtComment.eq(AdtComment)
34      and TheActor.rnInterfaceIN^ieMessage("Added comment to the crisis")}
35
36  /* Post Protocol:*/
37  postP{ true}
```

Listing 5.24: Messir (MCL-oriented) specification of the operation *oeReportOnCrisis*.

5.3.12 Operation Model for oeCloseCrisis

The `oeCloseCrisis` operation has the following properties:

Operation	
oeCloseCrisis	
sent to indicate that a crisis should be considered as closed.	
Parameters	
1	**AdtCrisisID: dtCrisisID** the identification information used to determine the crisis to close
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP 1 the system is started
PreP 2	PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
Pre-Condition (functional)	
PreF 1	PreF 1 it is supposed that there exist one ctCrisis instance with the same id attribute value as the one provided by the coordinator actor who wants to close.
Post-Condition (functional)	
PostF 1	PostF 1 the ctCrisis class instance having the provided id is considered closed in the post state.
PostF 2	PostF 2 There is no handler declared in the system as associated to the crisis.
PostF 3	PostF 3 all the alert instances associated to this crisis do not belong any more to the systems post state.
PostF 4	PostF 4 the coordinator actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	PostP 1 none

The listing 5.25 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actCoordinator in
5      let TheCrisis:ctCrisis in
6
7      self.rnActor.rnSystem = TheSystem
8      and self.rnActor = The Actor
9
10     /* Pre P01 */
11     TheSystem.vpStarted = true
```

```
12
13      /* Pre P02 */
14      TheActor.rnctAuthenticated.vpIsLogged = true
15
16      /* Pre P03 */
17      if(ctCrisis.type = huge)
18      then(
19       TheActor.accessRights = huge
20      )
21      else(
22       if(ctCrisis.type = medium)
23       then(
24        TheActor.accessRights = huge
25        or TheActor.accessRights = medium
26       )
27       else(
28        TheActor.accessRights = huge
29        or TheActor.accessRights = medium
30        or TheActor.accessRights = small
31       )
32       endif
33      )
34      endif}
35
36  /* Pre Functional:*/
37  preF{let TheSystem:ctState in
38      let ColctCrisis:Bag(ctCrisis) in
39
40      self.rnActor.rnSystem = TheSystem
41
42      /* Pre F01 */
43      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
44      and ColctCrisis->size.eq(1)}
45
46  /* Post Functional:*/
47  postF{let TheSystem:ctState in
48      let ThectCrisis:ctCrisis in
49
50      self.rnActor.rnSystem = TheSystem
51
52      /* Post F01 */
53      TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
54      and ThectCrisis.AetCrisisStatus.eq("closed")
55      and TheActor.rnInterfaceIN^ieMessage("Crisis closed")}
56
57  /* Post Protocol:*/
58  postP{ true}
```

Listing 5.25: Messir (MCL-oriented) specification of the operation *oeCloseCrisis*.

5.4 Environment - Out Interface Operation Scheme for actPerson

5.4.1 Operation Model for oeSearchPI

The `oeSearchPI` operation has the following properties:

Operation	
oeSearchPI	
sent to search for a specific PI in the system.	
Parameters	
1	**APIName: dtName**
	name used for ctPI instance retrieval
2	**APICategory: etCategory**

continues in next page . . .

... Operation table continuation

	category used for ctPI instance retrieval
3	**APICity: dtCity** city used for ctPI instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	none
Post-Condition (functional)	
PostF 1	PostF01 the actor ActPerson gets a response from the system if the PI has been found or not.
PostF 2	PostF02 following the response, the person actor is informed about the appropriated response of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.26 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actPerson in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{true}
17
18 /* Post Functional:*/
19 postF{let TheSystem:ctState in
20     let TheActor:actPerson in
21     let ColctPI:Set(ctPI) in
22
23     self.rnActor.rnSystem = TheSystem
24     and self.rnActor = TheActor
25
26     /* PostF01 */
27     TheSystem.rnctPI->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) = ColctPI
28     and if (ColctPI->size().eq(1))
29     then (
30      and AMessage.eq('PI found. How to proceed?')
31      and TheActor.rnInterfaceIN^ieMessage(AMessage)
32     )
33     else (
34      and AMessage.eq('PI NOT found. How to proceed?')
35      and TheActor.rnInterfaceIN^ieMessage(AMessage)
36     )
```

```
37      endif}
38
39  /* Post Protocol:*/
40  postP{ true}
```

Listing 5.26: Messir (MCL-oriented) specification of the operation *oeSearchPI*.

5.4.2 Operation Model for oeSendNewRequest

The `oeSendNewRequest` operation has the following properties:

OPERATION	
oeSendNewRequest	
sent to add a new request to the systems post state.	
Parameters	
1	**APIName: dtName** used to initialise the name field
2	**APICategory: etCategory** used to initialise the category field
3	**APICity: dtCity** used to initialise the city field
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there does NOT exists one ctRequest instance with the same name, category and city attributes than the one the person wants to add a new request.
PreF 2	PreF02 it is supposed that there does NOT exists one ctPI instance with the same name, category and city attributes than the one the person wants to add a new request.
Post-Condition (functional)	
PostF 1	PostF01 the new requested PI is added in the system's post state.
PostF 2	PostF02 the person actor is informed about the satisfaction of its request.
Post-Condition (protocol)	
PostP 1	none

The listing 5.27 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actPerson in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
```

```
13      and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15  /* Pre Functional:*/
16  preF{let TheSystem:ctState in
17      let TheActor:actPerson in
18      let ColctPI:Set(ctPI) in
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctPI->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) = ColctPI
25      and ColctPI->IsEmpty() = true
26
27      /* PreF02 */
28      and TheSystem.rnctRequest->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) =
            ColctRequest
29      and ColctRequest->IsEmpty() = true}
30
31  /* Post Functional:*/
32  postF{let TheSystem:ctState in
33      let TheActor:actPerson in
34      let ThectRequest:ctRequest in
35
36      self.rnActor.rnSystem = TheSystem
37      and self.rnActor = TheActor
38
39      /* PostF01 */
40      ThectRequest.init(APIID, APIName, APICategory, APICity)
41
42      /* PostF02 */
43      and AMessage.eq('Request sent!')
44      and TheActor.rnInterfaceIN^ieMessage(AMessage)}
45
46  /* Post Protocol:*/
47  postP{ true}
```

Listing 5.27: Messir (MCL-oriented) specification of the operation *oeSendNewRequest*.

5.4.3 Operation Model for oeGetGPSLocation

The `oeGetGPSLocation` operation has the following properties:

OPERATION	
oeGetGPSLocation	
sent to get the GPS location of a PI in the systems post state.	
Parameters	
1	**APIID: dtID** id used for ctPI instance retrieval
Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same id attribute than the one the person wants to get the gps location.
Post-Condition (functional)	
PostF 1	PostF01 the actor ActPerson gets an appropriate response with the gps location of the PI.

continues in next page . . .

... Operation table continuation

Post-Condition (protocol)	
PostP 1	none

The listing 5.28 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4      let TheActor:actPerson in
5
6      self.rnActor.rnSystem = TheSystem
7      and self.rnActor = TheActor
8
9      /* PreP01 */
10     TheSystem.vpStarted = true
11
12     /* PreP02 */
13     and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17     let TheActor:actPerson in
18     let ColctPI:Set(ctPI) in
19
20     self.rnActor.rnSystem = TheSystem
21     and self.rnActor = TheActor
22
23     /* PreF01 */
24     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
25     and ColctPI->IsEmpty() = false}
26
27 /* Post Functional:*/
28 postF{let TheSystem:ctState in
29     let TheActor:actPerson in
30
31     self.rnActor.rnSystem = TheSystem
32     and self.rnActor = TheActor
33
34     /* PostF01 */
35     AMessage.eq('Here, the GPS Location: 55.9797/56.9898')
36     and TheActor.rnInterfaceIN^ieMessage(AMessage)}
37
38 /* Post Protocol:*/
39 postP{ true}
```

Listing 5.28: Messir (MCL-oriented) specification of the operation *oeGetGPSLocation*.

5.4.4 Operation Model for oeGetDescription

The `oeGetDescription` operation has the following properties:

OPERATION	
oeGetDescription	
sent to get the description of a PI in the systems post state.	
Parameters	
1	**APIID: dtID**
	id used for ctPI instance retrieval

continues in next page ...

... Operation table continuation

Return type	
ptBoolean	
Pre-Condition (protocol)	
PreP 1	PreP01 the system is started.
PreP 2	PreP02 the actor logged previously and did not log out!
Pre-Condition (functional)	
PreF 1	PreF01 it is supposed that there exists one ctRequest instance with the same id attribute than the one the person wants to get the description.
Post-Condition (functional)	
PostF 1	PostF01 the actor ActPerson gets an appropriate response with the description of the PI.
Post-Condition (protocol)	
PostP 1	none

The listing 5.29 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Pre Protocol:*/
3  preP{let TheSystem:ctState in
4       let TheActor:actPerson in
5
6       self.rnActor.rnSystem = TheSystem
7       and self.rnActor = TheActor
8
9       /* PreP01 */
10      TheSystem.vpStarted = true
11
12      /* PreP02 */
13      and TheActor.rnctAuthenticated.vpIsLogged = true}
14
15 /* Pre Functional:*/
16 preF{let TheSystem:ctState in
17      let TheActor:actPerson in
18      let ColctPI:Set(ctPI) in
19
20      self.rnActor.rnSystem = TheSystem
21      and self.rnActor = TheActor
22
23      /* PreF01 */
24      TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
25      and ColctPI->IsEmpty() = false}
26
27 /* Post Functional:*/
28 postF{let TheSystem:ctState in
29      let TheActor:actPerson in
30
31      self.rnActor.rnSystem = TheSystem
32      and self.rnActor = TheActor
33
34      /* PostF01 */
35      AMessage.eq('Cactus Bascharage is a huge supermarket situated in the south of Luxembourg.')
36      and TheActor.rnInterfaceIN^ieMessage(AMessage)}
37
38 /* Post Protocol:*/
39 postP{ true}
```

Listing 5.29: Messir (MCL-oriented) specification of the operation *oeGetDescription*.

5.5 Environment - Actor Operation Schemes

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

5.6 Primary Types - Operation Schemes for Class ctAdministrator

5.6.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialise the current object as a new instance of the ctAdministrator type.	
Parameters	
1	**Alogin: dtLogin** the login initialised for a new administrator.
2	**Apwd: dtPassword** the password initialised for a new administrator.
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctAdministrator instance having its attributes equal to the ones provided as parameters.

The listing 5.30 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{if (
4       let Self:ctAdministrator in
5
6       /* Post F01 */
7        Self.login(Alogin)
8        and Self.pwd = Apwd
9        and Self.vpIsLogged = false
10
11       /* Post F02 */
12       and (Self.oclIsNew and self = Self)
13      )
14      then (result = true)
15      else (result = false)
16       endif}
```

Listing 5.30: Messir (MCL-oriented) specification of the operation *init*.

5.7 Primary Types - Operation Schemes for Class ctCoordinator

5.7.1 Operation Model for init

The `init` operation has the following properties:

OPERATION	
init	
used to initialise the current object as a new instance of the ctCoordinator type.	
Parameters	
1	**Aid: dtCoordinatorID** the id initialised for a new coordinator.
2	**Alogin: dtLogin** the login initialised for a new coordinator.
3	**Apwd: dtPassword** the password initialised for a new coordinator.
4	**Aaccessrights: etCrisisType** the access righta initialised for a new coordinator.
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctCoordinator instance having its attributes equal to the ones provided as parameters.

The listing 5.31 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{if (
4       let Self:ctCoordinator in
5
6       /* Post F01 */
7       Self.id = Aid
8       and Self.login = Alogin
9       and Self.pwd = Apwd
10      and Self.vpIsLogged = false
11      and Self.accessRights = AaccessRights
12
13      /* Post F02 */
14      and (Self.oclIsNew and self = Self)
15     )
16     then (result = true)
17     else (result = false)
18     endif}
```

Listing 5.31: Messir (MCL-oriented) specification of the operation *init*.

5.8 Primary Types - Operation Schemes for Class ctPerson

5.8.1 Operation Model for init

The `init` operation has the following properties:

OPERATION
init
used to initialise the current object as a new instance of the ctPerson type.
Parameters

continues in next page . . .

. . . Operation table continuation

1	**Aid: dtPhoneNumber** the id initialised for a new person.
2	**Alogin: dtLogin** the login initialised for a new PI.
3	**Apwd: dtPassword** the password initialised for a new PI.
4	**ApersonType: etPersonType** the person type initialised for a new PI.
Return type	
ptBoolean	
Post-Condition (functional)	
PostF 1	true iff the system poststate includes the current object as a new ctPerson instance having its attributes equal to the ones provided as parameters.

The listing 5.32 provides the Messir (MCL-oriented) specification of the operation.

```
1
2  /* Post Functional:*/
3  postF{if (
4       let Self:ctPerson in
5
6       /* Post F01 */
7
8       Self.id = Aid
9       and Self.login = Alogin
10      and Self.pwd = Apwd
11      and Self.type = Atype
12      and Self.phoneNumber = AphoneNumber
13
14       /* Post F02 */
15       and (Self.oclIsNew and self = Self)
16      )
17      then (result = true)
18      else (result = false)
19       endif}
```

Listing 5.32: Messir (MCL-oriented) specification of the operation *init*.

5.9 Primary Types - Operation Schemes for Datatypes

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

5.10 Primary Types - Operation Schemes for Enumerations

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

5.11 Secondary Types - Operation Schemes for Classes

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

5.12 Secondary Types - Operation Schemes for Datatypes

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

5.13 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

Messir Specification Files Listing

A.1 File ./src-gen/messir-spec/.views.msr

```
//
//DON'T TOUCH THIS FILE !!!
//
package uuid5703615e7218493bb67cdc3f6b2e26dc {
 Concept Model {}
}
```

Listing A.1: Messir Spec. file .views.msr.

A.2 File ./src-gen/messir-spec/operations/environment/actAdministrator.msr

```
/*
* @author mikel
* @date Fri Mar 16 11:46:38 CET 2018
*/

package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actAdministrator {

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

import lu.uni.lassy.excalibur.MyCrash.G02.environment
import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes

 Operation Model {

  operation: actAdministrator.outactAdministrator.oeAddPI(APIID:dtID, APIName:dtName, APICity:dtCity
      , APIGPSLocation:dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory):
      ptBoolean {

   /* Pre Protocol: */
   preP {
    let TheSystem:ctState in
    let TheActor:actAdministrator in

    self.rnActor.rnSystem = TheSystem
    and self.rnActor = TheActor

    /* PreP01 */
    TheSystem.vpStarted = true

    /* PreP02 */
    and TheActor.rnctAuthenticated.vpIsLogged = true
   }
```

```
35
36    /* Pre Functional: */
37    preF {
38     let TheSystem:ctState in
39     let TheActor:actAdministrator in
40     let ColctPI:Set(ctPI) in
41     let ColctRequest:Set(ctRequest) in
42
43     self.rnActor.rnSystem = TheSystem
44     and self.rnActor = TheActor
45
46     /* PreF01 */
47     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
48     and ColctPI->IsEmpty() = true
49
50     /* PreF02 */
51     and TheSystem.rnctRequest->select(id.eq(APIID) status.eq("solved")) = ColctRequest
52     and ColctRequest->IsEmpty() = false
53    }
54
55    /* Post Functional: */
56    postF {
57     let TheSystem:ctState in
58     let TheActor:actAdministrator in
59     let ThectPI:ctPI in
60     let ThePerson:actPerson in
61
62     self.rnActor.rnSystem = TheSystem
63     and self.rnActor = TheActor
64
65     /* PostF01 */
66     ThectPI.init(APIID, APIName, APICity, APIGPSLocation, APIDescription, APICategory)
67
68     /* PostF02 */
69     and TheActor.rnInterfaceIN^iePIAdded()
70
71     /* PostF03 */
72     and ThePerson.rnInterfaceIN^iePIAdded()
73    }
74
75    /* Post Protocol:*/
76    postP {
77     true
78    }
79   }
80
81   operation: actAdministrator.outactAdministrator.oeUpdatePI(APIID:dtID, APIName:dtName, APICity:
         dtCity, APIGPSLocation:dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory):
         ptBoolean {
82
83    /* Pre Protocol: */
84    preP {
85     let TheSystem:ctState in
86     let TheActor:actAdministrator in
87
88     self.rnActor.rnSystem = TheSystem
89     and self.rnActor = TheActor
90
91     /* PreP01 */
92     TheSystem.vpStarted = true
93
94     /* PreP02 */
95     and TheActor.rnctAuthenticated.vpIsLogged = true
96    }
97
98    /* Pre Functional: */
99    preF {
100    let TheSystem:ctState in
101    let TheActor:actAdministrator in
102    let ColctPI:Set(ctPI) in
```

```
103
104     self.rnActor.rnSystem = TheSystem
105     and self.rnActor = TheActor
106
107     /* PreF01 */
108     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
109     and ColctPI->size().eq(1)
110    }
111
112    /* Post Functional: */
113    postF {
114     let TheSystem:ctState in
115     let ThectPI:ctPI in
116     let TheActor:actAdministrator in
117
118     self.rnActor.rnSystem = TheSystem
119     and self.rnActor = TheActor
120
121     /* PostF01 */
122     ThectPI.rnctPI.name = APIName
123     and ThectPI.rnctPI.city = APIName
124     and ThectPI.rnctPI.location = AGPSLocation
125     and ThectPI.rnctPI.description = APIDescription
126     and ThectPI.rnctPI.category = APICategory
127
128     /* PostF02 */
129     and TheActor.rnInterfaceIN^iePIUpToDate()
130    }
131
132    /* Post Protocol:*/
133    postP {
134     true
135    }
136   }
137
138   operation: actAdministrator.outactAdministrator.oeDeletePI(APIID:dtID):ptBoolean {
139
140    /* Pre Protocol: */
141    preP {
142     let TheSystem:ctState in
143     let TheActor:actAdministrator in
144
145     self.rnActor.rnSystem = TheSystem
146     and self.rnActor = TheActor
147
148     /* PreP01 */
149     TheSystem.vpStarted = true
150
151     /* PreP02 */
152     and TheActor.rnctAuthenticated.vpIsLogged = true
153    }
154
155    /* Pre Functional: */
156    preF {
157     let TheSystem:ctState in
158     let TheActor:actAdministrator in
159     let ColctPI:Set(ctPI) in
160
161     self.rnActor.rnSystem = TheSystem
162     and self.rnActor = TheActor
163
164     /* PreF01 */
165     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
166     and ColctPI->size().eq(1)
167    }
168
169    /* Post Functional */
170    postF {
171     let TheSystem:ctState in
172     let TheActor:actAdministrator in
```

```
173     let ThectPI:ctPI in
174
175     self.rnActor.rnSystem = TheSystem
176     and self.rnActor = TheActor
177
178     /* PostF01 */
179     TheSystem.rnctPI->select(id.ed(APIID)) = ThectPI
180     and ThectPI.msrIsKilled
181
182     /* PostF02 */
183     and TheActor.rnInterfaceIN^iePIDeleted()
184    }
185
186    /* Post Protocol:*/
187    postP {
188     true
189    }
190   }
191
192   operation: actAdministrator.outactAdministrator.oeGetAllRequestsFromCoordinator():ptBoolean {
193
194    /* Pre Protocol: */
195    preP {
196     let TheSystem:ctState in
197     let TheActor:actAdministrator in
198
199     self.rnActor.rnSystem = TheSystem
200     and self.rnActor = TheActor
201
202     /* PreP01 */
203     TheSystem.vpStarted = true
204
205     /* PreP02 */
206     and TheActor.rnctAuthenticated.vpIsLogged = true
207    }
208
209    /* Pre Functional: */
210    preF {
211     true
212    }
213
214    /* Post Functional */
215    postF {
216     let TheSystem:ctState in
217     let TheActor:actAdministrator in
218     let ColctRequest:Set(ctRequest) in
219
220     self.rnActor.rnSystem = TheSystem
221     and self.rnActor = TheActor
222
223     TheSystem.rnctRequest->select(status.eq("pending")) = ColctRequest
224
225     and if (ColctRequest->IsEmpty() = false)
226     then (
227      /* PostF01 */
228      and TheActor.rnInterfaceIN^ieRequestList()
229     )
230     else (
231      /* PostF02 */
232      and AMessage.eq('No requested points of interest!')
233      and TheActor.rnInterfaceIN^ieMessage(AMessage)
234     )
235     endif
236    }
237
238    /* Post Protocol:*/
239    postP {
240     true
241    }
242   }
```

```
243
244  operation: actAdministrator.outactAdministrator.oeTreatRequest(ARequestID:dtID):ptBoolean {
245
246   /* Pre Protocol: */
247   preP {
248    let TheSystem:ctState in
249    let TheActor:actAdministrator in
250
251    self.rnActor.rnSystem = TheSystem
252    and self.rnActor = TheActor
253
254    /* PreP01 */
255    TheSystem.vpStarted = true
256
257    /* PreP02 */
258    and TheActor.rnctAuthenticated.vpIsLogged = true
259   }
260
261   /* Pre Functional: */
262   preF {
263    let TheSystem:ctState in
264    let TheActor:actAdministrator in
265    let ColctRequest:Set(ctRequest) in
266
267    self.rnActor.rnSystem = TheSystem
268    and self.rnActor = TheActor
269
270    /* PreF01 */
271    TheSystem.rnctRequest->select(status.eq("pending")) = ColctRequest
272    and ColctRequest->IsEmpty() = false
273   }
274
275   /* Post Functional */
276   postF {
277    let TheSystem:ctState in
278    let TheActor:actAdministrator in
279    let ThectRequest:ctRequest in
280
281    self.rnActor.rnSystem = TheSystem
282    and self.rnActor = TheActor
283
284    /* PostF01 */
285    ThectRequest.rnctRequest.status = "treated"
286
287    /* PostF02 */
288    and TheActor.rnInterfaceIN^ieRequestBeingTreated()
289   }
290
291   /* Post Protocol:*/
292   postP {
293    true
294   }
295  }
296
297  operation: actAdministrator.outactAdministrator.oeSolveRequest(ARequestID:dtID):ptBoolean {
298
299   /* Pre Protocol: */
300   preP {
301    let TheSystem:ctState in
302    let TheActor:actAdministrator in
303
304    self.rnActor.rnSystem = TheSystem
305    and self.rnActor = TheActor
306
307    /* PreP01 */
308    TheSystem.vpStarted = true
309
310    /* PreP02 */
311    and TheActor.rnctAuthenticated.vpIsLogged = true
312   }
```

```
313
314   /* Pre Functional: */
315   preF {
316    let TheSystem:ctState in
317    let TheActor:actAdministrator in
318    let ColctRequest:Set(ctRequest) in
319
320    self.rnActor.rnSystem = TheSystem
321    and self.rnActor = TheActor
322
323    /* PreF01 */
324    TheSystem.rnctRequest->select(status.eq("treated")) = ColctRequest
325    and ColctRequest->IsEmpty() = false
326   }
327
328   /* Post Functional */
329   postF {
330    let TheSystem:ctState in
331    let TheActor:actAdministrator in
332    let ThectRequest:ctRequest in
333
334    self.rnActor.rnSystem = TheSystem
335    and self.rnActor = TheActor
336
337    /* PostF01 */
338    ThectRequest.rnctRequest.status = "solved"
339
340    /* PostF02 */
341    and TheActor.rnInterfaceIN^ieRequestSolved()
342   }
343
344   /* Post Protocol:*/
345   postP {
346    true
347   }
348  }
349
350  operation: actAdministrator.outactAdministrator.oeAddCoordinator(AdtCoordinatorID: dtCoordinatorID
       , AdtLogin: dtLogin, AdtPassword: dtPassword, CoordinatorAccessRights: etCrisisType):ptBoolean
        {
351   /* Pre protocol */
352   preP {
353    let TheSystem:ctState in
354    let TheActor:actAdministrator in
355
356    self.rnActor.rnSystem = TheSystem
357    and self.rnActor = The Actor
358
359    /* Pre P01 */
360    TheSystem.vpStarted = true
361
362    /* Pre P02 */
363    TheActor.rnctAuthenticated.vpIsLogged = true
364   }
365
366   /* Pre Functional */
367   preF {
368    let TheSystem: ctState in
369    let TheActor:actAdministrator in
370    let ColctCoordinators:Bag(ctCoordinator) in
371
372    self.rnActor.rnSystem = TheSystem
373    and self.rnActor = TheActor
374
375    /* PreF01 */
376    and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
377    and ColctCoordinators->isEmpty() = true
378   }
379
380   /* Post Functional */
```

```
381   postF {
382    let TheSystem: ctState in
383    let TheactCoordinator:actCoordinator in
384    let ThectCoordinator:ctCoordinator in
385
386    self.rnActor.rnSystem = TheSystem
387    and self.rnActor = TheActor
388
389    /* PostF01 */
390    TheactCoordinator.init()
391
392    /* PostF02 */
393    and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword, CoordinatorAccessRights)
394
395    /* PostF03 */
396    and TheactCoordinator@post.rnctCoordinator = ThectCoordinator
397
398    /* PostF04 */
399    and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator
400
401    /* PostF05 */
402    and TheActor.rnInterfaceIN^ieCoordinatorAdded()
403   }
404
405   /* Post Protocol */
406   postP {
407    true
408   }
409  }
410
411  operation: actAdministrator.outactAdministrator.oeDeleteCoordinator(AdtCoordinatorID:
       dtCoordinatorID):ptBoolean {
412   /* Pre protocol */
413   preP {
414    let TheSystem:ctState in
415    let TheActor:actAdministrator in
416
417    self.rnActor.rnSystem = TheSystem
418    and self.rnActor = The Actor
419
420    /* Pre P01 */
421    TheSystem.vpStarted = true
422
423    /* Pre P02 */
424    TheActor.rnctAuthenticated.vpIsLogged = true
425   }
426
427   /* Pre Functional:*/
428   preF{
429    let TheSystem: ctState in
430    let TheActor:actAdministrator in
431    let ColctCoordinators:Bag(ctCoordinator) in
432
433    self.rnActor.rnSystem = TheSystem
434    and self.rnActor = TheActor
435
436    /* PreF01 */
437    TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
438    and ColctCoordinators->size().eq(1)
439   }
440
441   /* Post Functional:*/
442   postF{
443    let TheSystem: ctState in
444    let TheActor:actAdministrator in
445    let ThectCoordinator:ctCoordinator in
446
447    self.rnActor.rnSystem = TheSystem
448    and self.rnActor = TheActor
449
```

```
450     /* PostF01 */
451     TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ThectCoordinator
452     and ThectCoordinator.rnactCoordinator->forAll(msrIsKilled)
453     and ThectCoordinator.msrIsKilled
454
455     /* PostF02 */
456     and TheActor.rnInterfaceIN^ieCoordinatorDeleted()
457
458     /* Post Protocol:*/
459      * /* PostP01 */
460     and true
461    }
462
463    /* Post Protocol:*/
464    postP{
465     true
466    }
467   }
468
469   operation: actAdministrator.outactAdministrator.oeUpdateCoordinatorAccessRights(AdtCoordinatorID:
        dtCoordinatorID, CoordinatorAccessRights: etCrisisType):ptBoolean {
470    /* Pre protocol */
471    preP {
472     let TheSystem:ctState in
473     let TheActor:actAdministrator in
474
475     self.rnActor.rnSystem = TheSystem
476     and self.rnActor = The Actor
477
478     /* Pre P01 */
479     TheSystem.vpStarted = true
480
481     /* Pre P02 */
482     TheActor.rnctAuthenticated.vpIsLogged = true
483    }
484
485    /* Pre Functional:*/
486    preF{
487     let TheSystem: ctState in
488     let TheActor:actAdministrator in
489     let ColctCoordinators:Bag(ctCoordinator) in
490
491     self.rnActor.rnSystem = TheSystem
492     and self.rnActor = TheActor
493
494     /* PreF01 */
495     TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ColctCoordinators
496     and ColctCoordinators->size().eq(1)
497    }
498
499    /* Post Functional */
500    postF {
501     let TheSystem: ctState in
502     let TheActor:actAdministrator in
503     let ThectCoordinator:ctCoordinator in
504
505     self.rnActor.rnSystem = TheSystem
506     and self.rnActor = TheActor
507
508     /* PostF01 */
509     TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID)) = ThectCoordinators
510     and ThectCoordinator.accessRights.eq(AaccessRights)
511
512     /* PostF02 */
513     and TheActor.rnInterfaceIN^ieCoordinatorAccessRightsUpdated()
514    }
515
516    /* Post Protocol:*/
517    postP{
518     true
```

```
519    }
520   }
521  }
522 }
```

Listing A.2: Messir Spec. file actAdministrator.msr.

A.3 File ./src-gen/messir-spec/operations/environment/actAuthenticated.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actAuthenticated {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
15 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
16
17  Operation Model {
18
19   operation: actAuthenticated.outactAuthenticated.oeLogin(ALogin:dtLogin, APassword:dtPassword):
         ptBoolean{
20    /* Pre Protocol:*/
21    preP{let TheSystem: ctState in
22    let TheActor:actAuthenticated in
23    self.rnActor.rnSystem = TheSystem
24    and self.rnActor = TheActor
25
26    /* PreP01 */
27    and TheSystem.vpStarted = true
28    /* PreP02 */
29    and TheActor.rnctAuthenticated.vpIsLogged = false
30    /* PreP03 */
31    and TheActor.rnctAuthenticated.capReq = false}
32
33    /* Pre Functional:*/
34    preF{/* PreF01 */
35    true}
36
37    /* Post Functional:*/
38    postF{let TheSystem: ctState in
39    let TheactAuthenticated:actAuthenticated in
40
41    let AptStringMessageForTheactAuthenticated: ptString in
42    let AptStringMessageForTheactAdministrator:ptString in
43
44    self.rnActor.rnSystem = TheSystem
45    and self.rnActor = TheactAuthenticated
46
47    and /* PostF01 */
48     if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
49      and TheactAuthenticated.rnctAuthenticated.login = AdtLogin )
50     then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
51      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
52      )
53     else (AptStringMessageForTheactAuthenticated
54      .eq('Wrong identification information ! Please try again ...')
55      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
56      and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
57      and TheSystem.rnactAdministrator
58      .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
59      )
```

```
60     endif}
61
62    /* Post Protocol:*/
63    postP{ let TheSystem: ctState in
64    let TheactAuthenticated:actAuthenticated in
65
66    self.rnActor.rnSystem = TheSystem
67    and self.rnActor = TheactAuthenticated
68    /* PostP01 */
69    if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword and
70    TheactAuthenticated.rnctAuthenticated.login = AdtLogin) then(
71     TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true and
72     TheactAuthenticated.rnctAuthenticated@post.tries = 0 and
73     TheactAuthenticated.rnctAuthenticated@post.lastAccess = -181
74    )else(
75     TheactAuthenticated.rnctAuthenticated@post.lastAccess = ((int)System.currentTimeMillis()/1000) -
           TheactAuthenticated.rnctAuthenticated.lastAccess
76     if(TheactAuthenticated.rnctAuthenticated@post.lastAccess <= 180)then(
77      TheactAuthenticated.rnctAuthenticated@post.tries = TheactAuthenticated.rnctAuthenticated@post.
            tries + 1
78      if (TheactAuthenticated.rnctAuthenticated@post.tries > 2) then(
79       TheactAuthenticated.rnctAuthenticated@post.capReq = true
80      )else
81       true
82      endif
83     )else(
84      TheactAuthenticated.rnctAuthenticated@post.tries = 1
85     )endif
86    )endif}
87   }
88
89   operation: actAuthenticated.outactAuthenticated.oeLogout():ptBoolean{
90   // include below the specification information (pre,post or ocl or prolog)
91    /* Pre Protocol:*/
92    preP{let TheSystem: ctState in
93    let TheActor:actAdministrator in
94    self.rnActor.rnSystem = TheSystem
95    and self.rnActor = TheActor
96
97    /* PreP01 */
98    and TheSystem.vpStarted = true
99     /* PreP02 */
100   and TheActor.rnctAuthenticated.vpIsLogged = true}
101
102   /* Pre Functional:*/
103   preF{/* PreF01 */
104   true}
105
106   /* Post Functional:*/
107   postF{let TheSystem: ctState in
108   let TheactAuthenticated:actAuthenticated in
109   let AptStringMessageForTheactAuthenticated: ptString in
110
111   self.rnActor.rnSystem = TheSystem
112   and self.rnActor = TheactAuthenticated
113
114   /* PostF01 */
115   AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
116   and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)}
117
118   /* Post Protocol:*/
119   postP{ let TheSystem: ctState in
120   let TheactAuthenticated:actAuthenticated in
121
122   self.rnActor.rnSystem = TheSystem
123   and self.rnActor = TheactAuthenticated.asSet
124   /* PostP01 */
125   TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false}
126  }
127  operation: actAuthenticated.outactAuthenticated.oeLoginWithCaptcha(ALogin:dtLogin, APassword:
```

```
      dtPassword, ACaptcha:dtCaptcha):ptBoolean{
128   /* Pre Protocol:*/
129   preP{let TheSystem: ctState in
130   let TheActor:actAuthenticated in
131   self.rnActor.rnSystem = TheSystem
132   and self.rnActor = TheActor
133
134   /* PreP01 */
135   and TheSystem.vpStarted = true
136   /* PreP02 */
137   and TheActor.rnctAuthenticated.vpIsLogged = false
138   /* PreP03 */
139   and TheActor.rnctAuthenticated.capReq = true}
140
141   /* Pre Functional:*/
142   preF{/* PreF01 */
143   true}
144
145   /* Post Functional:*/
146   postF{let TheSystem: ctState in
147   let TheactAuthenticated:actAuthenticated in
148
149   let AptStringMessageForTheactAuthenticated:ptString in
150   let AptStringMessageForTheactAdministrator:ptString in
151
152   self.rnActor.rnSystem = TheSystem
153   and self.rnActor = TheactAuthenticated
154
155   and /* PostF01 */
156   if(ACaptcha = TheactAuthenticated.rnctAuthenticated.cap2Solve) then
157    if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
158     and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
159     )
160    then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
161     and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
162     )
163    else (AptStringMessageForTheactAuthenticated
164     .eq('Wrong identification information ! Please try again ...')
165     and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
166     and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
167     and TheSystem.rnactAdministrator
168     .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
169     )
170    endif
171   else
172    (TheactAuthenticated.rnctAuthenticated@post.cap2Solve = random() and
173     AptStringMessageForTheactAuthenticated.eq('Wrong Captcha information ! Captcha: ' +
174      TheactAuthenticated.rnctAuthenticated@post.cap2Solve) and
175     TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
176    )
177   endif}
178
179   /* Post Protocol:*/
180   postP{ let TheSystem: ctState in
181   let TheactAuthenticated:actAuthenticated in
182
183   self.rnActor.rnSystem = TheSystem
184   and self.rnActor = TheactAuthenticated
185   /* PostP01 */
186   if(ACaptcha = TheactAuthenticated.rnctAuthenticated.cap2Solve) then
187    (if(TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword and
188    TheactAuthenticated.rnctAuthenticated.login = AdtLogin) then(
189     TheactAuthenticated.rnctAuthenticated@post.tries = 0 and
190     TheactAuthenticated.rnctAuthenticated@post.lastAccess = -181 and
191     TheactAuthenticated.rnctAuthenticated@post.capReq = false and
192     TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true
193    )else(
194     TheactAuthenticated.rnctAuthenticated@post.lastAccess = ((int)System.currentTimeMillis()/1000)
          - TheactAuthenticated.rnctAuthenticated.lastAccess
195     (if(TheactAuthenticated.rnctAuthenticated@post.lastAccess <= 180)then(
```

```
196       TheactAuthenticated.rnctAuthenticated@post.tries = TheactAuthenticated.rnctAuthenticated.tries
              + 1
197      )else(
198       TheactAuthenticated.rnctAuthenticated@post.tries = 1
199      )endif)
200     )endif)
201    else
202     (false)
203    endif}
204   }
205   operation: actAuthenticated.outactAuthenticated.oeResetPassword(ALogin:dtLogin):ptBoolean{
206    /* Pre Protocol:*/
207    preP{let TheSystem: ctState in
208    let TheActor:actAuthenticated in
209    self.rnActor.rnSystem = TheSystem
210    and self.rnActor = TheActor
211
212    /* PreP01 */
213    and TheSystem.vpStarted = true
214    /* PreP02 */
215    and TheActor.rnctAuthenticated.vpIsLogged = false}
216
217    /* Pre Functional:*/
218    preF{/* PreF01 */
219    true}
220
221    /* Post Functional:*/
222    postF{let TheSystem: ctState in
223    let TheactAuthenticated:actAuthenticated in
224
225    let AptStringMessageForTheactAuthenticated: ptString in
226    let AptStringMessageForTheactAdministrator:ptString in
227
228    self.rnActor.rnSystem = TheSystem
229    and self.rnActor = TheactAuthenticated
230
231    and /* PostF01 */
232     if (TheactAuthenticated.rnctAuthenticated.login = AdtLogin
233      )
234     then (TheactAuthenticated.rnctAuthenticated@post.pwd = new dtPassword( random() )
235      and AptStringMessageForTheactAuthenticated.eq('User found, resetting password ... New password
             = ' + TheactAuthenticated.rnctAuthenticated@post.pwd)
236      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
237      )
238     else (AptStringMessageForTheactAuthenticated
239      .eq('Wrong username information ! Please try again ...')
240      and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
241      )
242     endif}
243
244    /* Post Protocol:*/
245    postP{
246     }
247   }
248  }
249 }
```

Listing A.3: Messir Spec. file actAuthenticated.msr.

A.4 File ./src-gen/messir-spec/operations/environment/actComCompany.m

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actComCompany {
7
8 import lu.uni.lassy.messir.libraries.calendar
```

```
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Operation Model {
14
15  }
16 }
```

Listing A.4: Messir Spec. file actComCompany.msr.

A.5 File ./src-gen/messir-spec/operations/environment/actCoordinator.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actCoordinator {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
15 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
16
17  Operation Model {
18
19   operation: actCoordinator.outactCoordinator.oeGetAllRequests():ptBoolean {
20
21    /* Pre Protocol: */
22    preP {
23     let TheSystem:ctState in
24     let TheActor:actAdministrator in
25
26     self.rnActor.rnSystem = TheSystem
27     and self.rnActor = TheActor
28
29     /* PreP01 */
30     TheSystem.vpStarted = true
31
32     /* PreP02 */
33     and TheActor.rnctAuthenticated.vpIsLogged = true
34
35     /* Pre P03 */
36     if(ctCrisis.type = huge)
37     then(
38      TheActor.accessRights = huge
39     )
40     else(
41      if(ctCrisis.type = medium)
42      then(
43       TheActor.accessRights = huge
44       or TheActor.accessRights = medium
45      )
46      else(
47       TheActor.accessRights = huge
48       or TheActor.accessRights = medium
49       or TheActor.accessRights = small
50      )
51      endif
52     )
53     endif
54    }
55
56    /* Pre Functional: */
```

```
57    preF {
58     true
59    }
60
61    /* Post Functional */
62    postF {
63     let TheSystem:ctState in
64     let TheActor:actCoordinator in
65     let ColctRequest:Set(ctRequest) in
66
67     self.rnActor.rnSystem = TheSystem
68     and self.rnActor = TheActor
69
70     TheSystem.rnctRequest->select(status.eq("")) = ColctRequest
71
72     and if (ColctRequest->IsEmpty() = false)
73     then (
74      /* PostF01 */
75      and TheActor.rnInterfaceIN^ieRequestListToCheck()
76     )
77     else (
78      /* PostF02 */
79      and AMessage.eq('No requests to check!')
80      and TheActor.rnInterfaceIN^ieMessage(AMessage)
81     )
82     endif
83    }
84
85    /* Post Protocol:*/
86    postP {
87     true
88    }
89   }
90
91   operation: actCoordinator.outactCoordinator.oeCheckAvailability(ARequestID:dtID):ptBoolean {
92
93    /* Pre Protocol: */
94    preP {
95     let TheSystem:ctState in
96     let TheActor:actAdministrator in
97
98     self.rnActor.rnSystem = TheSystem
99     and self.rnActor = TheActor
100
101    /* PreP01 */
102    TheSystem.vpStarted = true
103
104    /* PreP02 */
105    and TheActor.rnctAuthenticated.vpIsLogged = true
106   }
107
108   /* Pre Functional: */
109   preF {
110    let TheSystem:ctState in
111    let TheActor:actCoordinator in
112    let ColctPI:Set(ctPI) in
113    let ColctRequest:Set(ctRequest) in
114
115    self.rnActor.rnSystem = TheSystem
116    and self.rnActor = TheActor
117
118    /* PreF01 */
119    TheSystem.rnctRequest->select(name.eq(ARequestID)) = ColctRequest
120    and ColctRequest->size().eq(1)
121   }
122
123   /* Post Functional */
124   postF {
125    let TheSystem:ctState in
126    let TheActor:actCoordinator in
```

```
127     let ThectRequest:ctRequest in
128
129     self.rnActor.rnSystem = TheSystem
130     and self.rnActor = TheActor
131
132     TheSystem.rnctPI->select(name.eq(ARequestID)) = ColctPI
133
134     and if (ColctPI.IsEmpty() = false)
135     then (
136      /* PostF01 */
137      and ThectRequest.rnctRequest.ignored = true
138      and AMessage.eq('The PI is already in the system.')
139      and TheActor.rnInterfaceIN^ieMessage(AMessage)
140     )
141     else (
142      /* PostF02 */
143      and ThectRequest.rnctRequest.ignored = false
144      and AMessage.eq('The PI is NOT in the system. How to proceed?')
145      and TheActor.rnInterfaceIN^ieMessage(AMessage)
146     )
147     endif
148    }
149
150    /* Post Protocol:*/
151    postP {
152     true
153    }
154   }
155
156   operation: actCoordinator.outactCoordinator.oeDeliverRequest(ARequestID:dtID):ptBoolean {
157
158    /* Pre Protocol: */
159    preP {
160     let TheSystem:ctState in
161     let TheActor:actCoordinator in
162
163     self.rnActor.rnSystem = TheSystem
164     and self.rnActor = TheActor
165
166     /* PreP01 */
167     TheSystem.vpStarted = true
168
169     /* PreP02 */
170     and TheActor.rnctAuthenticated.vpIsLogged = true
171    }
172
173    /* Pre Functional: */
174    preF {
175     let TheSystem:ctState in
176     let TheActor:actCoordinator in
177     let ColctRequest:Set(ctRequest) in
178
179     self.rnActor.rnSystem = TheSystem
180     and self.rnActor = TheActor
181
182     /* PreF01 */
183     TheSystem.rnctRequest->select(ignored.eq(false)) = ColctRequest
184     and ColctRequest->IsEmpty() = false
185    }
186
187    /* Post Functional */
188    postF {
189     let TheSystem:ctState in
190     let TheActor:actCoordinator in
191     let ThectRequest:ctRequest in
192
193     self.rnActor.rnSystem = TheSystem
194     and self.rnActor = TheActor
195
196     /* PostF01 */
```

```
197     ThectRequest.rnctRequest.status = "pending"
198
199     /* PostF02 */
200     and TheActor.rnInterfaceIN^ieRequestDelivered()
201    }
202
203    /* Post Protocol:*/
204    postP {
205     true
206    }
207   }
208
209   operation: actCoordinator.outactCoordinator.oeValidateAlert(AdtAlertID: dtAlertID):ptBoolean {
210    /* Pre protocol */
211    preP {
212     let TheSystem:ctState in
213     let TheActor:actCoordinator in
214
215     self.rnActor.rnSystem = TheSystem
216     and self.rnActor = The Actor
217
218     /* Pre P01 */
219     TheSystem.vpStarted = true
220
221     /* Pre P02 */
222     TheActor.rnctAuthenticated.vpIsLogged = true
223    }
224
225    /* Pre Functional */
226    preF {
227     let TheSystem:ctState in
228     let ColctAlert:Bag(ctAlert) in
229
230     self.rnActor.rnSystem = TheSystem
231
232     /* Pre F01 */
233     TheSystem.rnctAlert->select(id.eq(AdtAlertID)) = ColctAlert
234     and ColctAlert->size.eq(1)
235    }
236
237    /* Post Functional */
238    postF {
239     let TheSystem:ctState in
240     let ThectAlert:ctAlert in
241
242     self.rnActor.rnSystem = TheSystem
243
244     /* Post F01 */
245     TheSystem.rnctAlert->select(id.eq(AdtAlerID)) = ThectAlert
246     and ThectAlert.status.eq("valid")
247     and TheActor.rnInterfaceIN^ieMessage("Alert validated!")
248    }
249
250    /* Post Protocol */
251    postP {
252     true
253    }
254   }
255
256   operation: actCoordinator.outactCoordinator.oeInvalidateAlert(AdtAlertID: dtAlertID):ptBoolean {
257    /* Pre protocol */
258    preP {
259     let TheSystem:ctState in
260     let TheActor:actCoordinator in
261
262     self.rnActor.rnSystem = TheSystem
263     and self.rnActor = The Actor
264
265     /* Pre P01 */
266     TheSystem.vpStarted = true
```

```
267
268     /* Pre P02 */
269     TheActor.rnctAuthenticated.vpIsLogged = true
270    }
271
272    /* Pre Functional */
273    preF {
274     let TheSystem:ctState in
275     let ColctAlert:Bag(ctAlert) in
276
277     self.rnActor.rnSystem = TheSystem
278
279     /* Pre F01 */
280     TheSystem.rnctAlert->select(id.eq(AdtAlertID)) = ColctAlert
281     and ColctAlert->size.eq(1)
282    }
283
284    /* Post Functional */
285    postF {
286     let TheSystem:ctState in
287     let ThectAlert:ctAlert in
288
289     self.rnActor.rnSystem = TheSystem
290
291     /* Post F01 */
292     TheSystem.rnctAlert->select(id.eq(AdtAlerID)) = ThectAlert
293     and ThectAlert.status.eq("closed")
294     and TheActor.rnInterfaceIN^ieMessage("Alert invalidated!")
295    }
296
297    /* Post Protocol */
298    postP {
299     true
300    }
301   }
302
303   operation: actCoordinator.outactCoordinator.oeGetAlertSet(AetAlertStatus: etAlertStatus):ptBoolean
         {
304    /* Pre protocol */
305    preP {
306     let TheSystem:ctState in
307     let TheActor:actCoordinator in
308
309     self.rnActor.rnSystem = TheSystem
310     and self.rnActor = The Actor
311
312     /* Pre P01 */
313     TheSystem.vpStarted = true
314
315     /* Pre P02 */
316     TheActor.rnctAuthenticated.vpIsLogged = true
317    }
318
319    /* Pre Functional */
320    preF {
321     true
322    }
323
324    /* Post Functional */
325    postF{
326     let TheSystem: ctState in
327     let ThectAlert:ctAlert in
328
329     self.rnActor.rnSystem = TheSystem
330
331     /* Post F01 */
332     TheSystem.rnctAlert->select(status.eq(AetAlertStatus)) = ThectAlert
333     and ThectAlert.rnctAlert->forAll(isSentToCoordinator)
334     and ThectAlert.msrIsSentToCoordinator
335    }
```

```
336
337   /* Post Protocol */
338   postP {
339    true
340   }
341  }
342
343  operation: actCoordinator.outactCoordinator.oeGetCrisisSet(AetCrisisStatus: etCrisisStatus):
         ptBoolean {
344   /* Pre protocol */
345   preP {
346    let TheSystem:ctState in
347    let TheActor:actCoordinator in
348
349    self.rnActor.rnSystem = TheSystem
350    and self.rnActor = The Actor
351
352    /* Pre P01 */
353    TheSystem.vpStarted = true
354
355    /* Pre P02 */
356    TheActor.rnctAuthenticated.vpIsLogged = true
357
358    /* Pre P03 */
359    /* Coordinators access rights are equal to or lower than the crisis type.*/
360   }
361
362   /* Pre Functional */
363   preF {
364    true
365   }
366
367   /* Post Functional */
368   postF{
369    let TheSystem: ctState in
370    let ThectCrisis:ctCrisis in
371
372    self.rnActor.rnSystem = TheSystem
373
374    /* Post F01 */
375    TheSystem.rnctCrisis->select(status.eq(AetCrisisStatus)) = ThectCrisis
376    and ThectCrisis.rnctCrisis->forAll(isSentToCoordinator)
377    and ThectCrisis.msrIsSentToCoordinator
378   }
379
380   /* Post Protocol */
381   postP {
382    true
383   }
384  }
385
386  operation: actCoordinator.outactCoordinator.oeSetCrisisType(AdtCrisisID: dtCrisisID, AetCrisisType
         : etCrisisType):ptBoolean {
387   /* Pre protocol */
388   preP {
389    let TheSystem:ctState in
390    let TheActor:actCoordinator in
391
392    self.rnActor.rnSystem = TheSystem
393    and self.rnActor = The Actor
394
395    /* Pre P01 */
396    TheSystem.vpStarted = true
397
398    /* Pre P02 */
399    TheActor.rnctAuthenticated.vpIsLogged = true
400
401    /* Pre P03 */
402    /* Coordinators access rights are equal to or lower than the crisis type.*/
403   }
```

```
404
405   /* PreFunctional */
406   preF {
407    let TheSystem:ctState in
408    let ColctCrisis:Bag(ctCrisis) in
409
410    self.rnActor.rnSystem = TheSystem
411
412    /* Pre F01 */
413    TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
414    and ColctCrisis->size.eq(1)
415   }
416
417   /* Post Functional */
418   postF {
419    let TheSystem:ctState in
420    let ThectCrisis:ctCrisis in
421
422    self.rnActor.rnSystem = TheSystem
423
424    /* Post F01 */
425    TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
426    and ThectCrisis.AetCrisisType.eq(AdtCrisisType)
427    and TheActor.rnInterfaceIN^ieMessage("Crisis type set to " + AdtCrisisType)
428   }
429
430   /* Post Protocol */
431   postP {
432    true
433   }
434  }
435
436  operation: actCoordinator.outactCoordinator.oeSetCrisisStatus(AdtCrisisID: dtCrisisID,
       AetCrisisStatus: etCrisisStatus):ptBoolean {
437   /* Pre protocol */
438   preP {
439    let TheSystem:ctState in
440    let TheActor:actCoordinator in
441
442    self.rnActor.rnSystem = TheSystem
443    and self.rnActor = The Actor
444
445    /* Pre P01 */
446    TheSystem.vpStarted = true
447
448    /* Pre P02 */
449    TheActor.rnctAuthenticated.vpIsLogged = true
450
451    /* Pre P03 */
452    /* Coordinators access rights are equal to or lower than the crisis type.*/
453   }
454
455   /* PreFunctional */
456   preF {
457    let TheSystem:ctState in
458    let ColctCrisis:Bag(ctCrisis) in
459
460    self.rnActor.rnSystem = TheSystem
461
462    /* Pre F01 */
463    TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
464    and ColctCrisis->size.eq(1)
465   }
466
467   /* Post Functional */
468   postF {
469    let TheSystem:ctState in
470    let ThectCrisis:ctCrisis in
471
472    self.rnActor.rnSystem = TheSystem
```

```
473
474     /* Post F01 */
475     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
476     and ThectCrisis.AetCrisisStatus.eq(AdtCrisisStatus)
477     and TheActor.rnInterfaceIN^ieMessage("Crisis status set to " + AdtCrisisStatus)
478    }
479
480    /* Post Protocol */
481    postP {
482     true
483    }
484   }
485
486   operation: actCoordinator.outactCoordinator.oeSetCrisisHandler(AdtCrisisID: dtCrisisID):ptBoolean
          {
487    /* Pre protocol */
488    preP {
489     let TheSystem:ctState in
490     let TheActor:actCoordinator in
491
492     self.rnActor.rnSystem = TheSystem
493     and self.rnActor = The Actor
494
495     /* Pre P01 */
496     TheSystem.vpStarted = true
497
498     /* Pre P02 */
499     TheActor.rnctAuthenticated.vpIsLogged = true
500
501     /* Pre P03 */
502     /* Coordinators access rights are equal to or lower than the crisis type.*/
503    }
504
505    /* PreFunctional */
506    preF {
507     let TheSystem:ctState in
508     let ColctCrisis:Bag(ctCrisis) in
509
510     self.rnActor.rnSystem = TheSystem
511
512     /* Pre F01 */
513     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
514     and ColctCrisis->size.eq(1)
515    }
516
517    /* Post Functional */
518    postF {
519     let TheSystem:ctState in
520     let ThectCrisis:ctCrisis in
521
522     self.rnActor.rnSystem = TheSystem
523
524     /* Post F01 */
525     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
526     and ThectCrisis.AetCrisisStatus.eq("handled")
527     and TheActor.rnInterfaceIN^ieMessage("Handler of the crisis " + AdtCrisisID + " is being handled
           by coordinator with id " + self.id)
528
529     /* Post F02 */
530     /*All the alerts related to this crisis are sent to the actor such that he can
531     decide how to handle them.*/
532     /* Post F03 */
533     /*if the crisis was already handled at pre-sate then the associated handler actor is
534     notified about the change of handler for one of his crisis
535     (n.b. it might be the same even if not relevant).*/
536     /* Post F04 */
537     /*a message is sent to the communication company for any human related to an alert
538     associated to the crisis. A human will receive as many messages as alerts he sent despite
539     the fact that they might relate to the same crisis (i.e. one alert, one acknoledgement).*/
540    }
```

```
541
542   /* Post Protocol */
543   postP {
544    true
545   }
546  }
547
548  operation: actCoordinator.outactCoordinator.oeReportOnCrisis(AdtCrisisID: dtCrisisID, AdtComment:
         dtComment):ptBoolean {
549   /* Pre protocol */
550   preP {
551    let TheSystem:ctState in
552    let TheActor:actCoordinator in
553
554    self.rnActor.rnSystem = TheSystem
555    and self.rnActor = The Actor
556
557    /* Pre P01 */
558    TheSystem.vpStarted = true
559
560    /* Pre P02 */
561    TheActor.rnctAuthenticated.vpIsLogged = true
562
563    /* Pre P03 */
564    /* Coordinators access rights are equal to or lower than the crisis type.*/
565   }
566
567   /* PreFunctional */
568   preF {
569    let TheSystem:ctState in
570    let ColctCrisis:Bag(ctCrisis) in
571
572    self.rnActor.rnSystem = TheSystem
573
574    /* Pre F01 */
575    TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
576    and ColctCrisis->size.eq(1)
577   }
578
579   /* Post Functional */
580   postF {
581    let TheSystem:ctState in
582    let ThectCrisis:ctCrisis in
583
584    self.rnActor.rnSystem = TheSystem
585
586    /* Post F01 */
587    TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
588    and ThectCrisis.AdtComment.eq(AdtComment)
589    and TheActor.rnInterfaceIN^ieMessage("Added comment to the crisis")
590   }
591
592   /* Post Protocol */
593   postP {
594    true
595   }
596  }
597
598  operation: actCoordinator.outactCoordinator.oeCloseCrisis(AdtCrisisID: dtCrisisID):ptBoolean {
599   /* Pre protocol */
600   preP {
601    let TheSystem:ctState in
602    let TheActor:actCoordinator in
603    let TheCrisis:ctCrisis in
604
605    self.rnActor.rnSystem = TheSystem
606    and self.rnActor = The Actor
607
608    /* Pre P01 */
609    TheSystem.vpStarted = true
```

```
610
611     /* Pre P02 */
612     TheActor.rnctAuthenticated.vpIsLogged = true
613
614     /* Pre P03 */
615     if(ctCrisis.type = huge)
616     then(
617      TheActor.accessRights = huge
618     )
619     else(
620      if(ctCrisis.type = medium)
621      then(
622       TheActor.accessRights = huge
623       or TheActor.accessRights = medium
624      )
625      else(
626       TheActor.accessRights = huge
627       or TheActor.accessRights = medium
628       or TheActor.accessRights = small
629      )
630      endif
631     )
632     endif
633    }
634
635    /* Pre Functional */
636    preF {
637     let TheSystem:ctState in
638     let ColctCrisis:Bag(ctCrisis) in
639
640     self.rnActor.rnSystem = TheSystem
641
642     /* Pre F01 */
643     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ColctCrisis
644     and ColctCrisis->size.eq(1)
645    }
646
647    /* Post Functional */
648    postF {
649     let TheSystem:ctState in
650     let ThectCrisis:ctCrisis in
651
652     self.rnActor.rnSystem = TheSystem
653
654     /* Post F01 */
655     TheSystem.rnctCrisis->select(id.eq(AdtCrisisID)) = ThectCrisis
656     and ThectCrisis.AetCrisisStatus.eq("closed")
657     and TheActor.rnInterfaceIN^ieMessage("Crisis closed")
658    }
659
660    /* Post Protocol */
661    postP {
662     true
663    }
664   }
665  }
666 }
```

Listing A.5: Messir Spec. file actCoordinator.msr.

A.6 File ./src-gen/messir-spec/operations/environment/actMsrCreator.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actMsrCreator {
7
```

```
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Operation Model {
14
15  }
16 }
```

Listing A.6: Messir Spec. file actMsrCreator.msr.

A.7 File ./src-gen/messir-spec/operations/environment/actPerson.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.actPerson {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
15 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
16
17  Operation Model {
18
19   operation: actPerson.outactPerson.oeSearchPI(APIName:dtName, APICategory:etCategory, APICity:
        dtCity):ptBoolean {
20
21    /* Pre Protocol: */
22    preP {
23     let TheSystem:ctState in
24     let TheActor:actPerson in
25
26     self.rnActor.rnSystem = TheSystem
27     and self.rnActor = TheActor
28
29     /* PreP01 */
30     TheSystem.vpStarted = true
31
32     /* PreP02 */
33     and TheActor.rnctAuthenticated.vpIsLogged = true
34    }
35
36    /* Pre Functional: */
37    preF {
38     true
39    }
40
41    /* Post Functional: */
42    postF {
43     let TheSystem:ctState in
44     let TheActor:actPerson in
45     let ColctPI:Set(ctPI) in
46
47     self.rnActor.rnSystem = TheSystem
48     and self.rnActor = TheActor
49
50     /* PostF01 */
51     TheSystem.rnctPI->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) = ColctPI
52     and if (ColctPI->size().eq(1))
53     then (
54      and AMessage.eq('PI found. How to proceed?')
```

```
55      and TheActor.rnInterfaceIN^ieMessage(AMessage)
56     )
57     else (
58      and AMessage.eq('PI NOT found. How to proceed?')
59      and TheActor.rnInterfaceIN^ieMessage(AMessage)
60     )
61     endif
62    }
63
64    /* Post Protocol:*/
65    postP {
66     true
67    }
68   }
69
70   operation: actPerson.outactPerson.oeSendNewRequest(APIID:dtID, APIName:dtName, APICategory:
         etCategory, APICity:dtCity):ptBoolean {
71
72    /* Pre Protocol: */
73    preP {
74     let TheSystem:ctState in
75     let TheActor:actPerson in
76
77     self.rnActor.rnSystem = TheSystem
78     and self.rnActor = TheActor
79
80     /* PreP01 */
81     TheSystem.vpStarted = true
82
83     /* PreP02 */
84     and TheActor.rnctAuthenticated.vpIsLogged = true
85    }
86
87    /* Pre Functional: */
88    preF {
89     let TheSystem:ctState in
90     let TheActor:actPerson in
91     let ColctPI:Set(ctPI) in
92
93     self.rnActor.rnSystem = TheSystem
94     and self.rnActor = TheActor
95
96     /* PreF01 */
97     TheSystem.rnctPI->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) = ColctPI
98     and ColctPI->IsEmpty() = true
99
100    /* PreF02 */
101    and TheSystem.rnctRequest->select(name.eq(APIName) category.eq(APICategory) city.eq(APICity)) =
         ColctRequest
102    and ColctRequest->IsEmpty() = true
103   }
104
105   /* Post Functional: */
106   postF {
107    let TheSystem:ctState in
108    let TheActor:actPerson in
109    let ThectRequest:ctRequest in
110
111    self.rnActor.rnSystem = TheSystem
112    and self.rnActor = TheActor
113
114    /* PostF01 */
115    ThectRequest.init(APIID, APIName, APICategory, APICity)
116
117    /* PostF02 */
118    and AMessage.eq('Request sent!')
119    and TheActor.rnInterfaceIN^ieMessage(AMessage)
120   }
121
122   /* Post Protocol:*/
```

```
123    postP {
124     true
125    }
126   }
127
128   operation: actPerson.outactPerson.oeGetGPSLocation(APIID:dtID):ptBoolean {
129
130    /* Pre Protocol: */
131    preP {
132     let TheSystem:ctState in
133     let TheActor:actPerson in
134
135     self.rnActor.rnSystem = TheSystem
136     and self.rnActor = TheActor
137
138     /* PreP01 */
139     TheSystem.vpStarted = true
140
141     /* PreP02 */
142     and TheActor.rnctAuthenticated.vpIsLogged = true
143    }
144
145    /* Pre Functional: */
146    preF {
147     let TheSystem:ctState in
148     let TheActor:actPerson in
149     let ColctPI:Set(ctPI) in
150
151     self.rnActor.rnSystem = TheSystem
152     and self.rnActor = TheActor
153
154     /* PreF01 */
155     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
156     and ColctPI->IsEmpty() = false
157    }
158
159    /* Post Functional: */
160    postF {
161     let TheSystem:ctState in
162     let TheActor:actPerson in
163
164     self.rnActor.rnSystem = TheSystem
165     and self.rnActor = TheActor
166
167     /* PostF01 */
168     AMessage.eq('Here, the GPS Location: 55.9797/56.9898')
169     and TheActor.rnInterfaceIN^ieMessage(AMessage)
170    }
171
172    /* Post Protocol:*/
173    postP {
174     true
175    }
176   }
177
178   operation: actPerson.outactPerson.oeGetDescription(APIID:dtID):ptBoolean {
179
180    /* Pre Protocol: */
181    preP {
182     let TheSystem:ctState in
183     let TheActor:actPerson in
184
185     self.rnActor.rnSystem = TheSystem
186     and self.rnActor = TheActor
187
188     /* PreP01 */
189     TheSystem.vpStarted = true
190
191     /* PreP02 */
192     and TheActor.rnctAuthenticated.vpIsLogged = true
```

```
193    }
194
195    /* Pre Functional: */
196    preF {
197     let TheSystem:ctState in
198     let TheActor:actPerson in
199     let ColctPI:Set(ctPI) in
200
201     self.rnActor.rnSystem = TheSystem
202     and self.rnActor = TheActor
203
204     /* PreF01 */
205     TheSystem.rnctPI->select(id.eq(APIID)) = ColctPI
206     and ColctPI->IsEmpty() = false
207    }
208
209    /* Post Functional: */
210    postF {
211     let TheSystem:ctState in
212     let TheActor:actPerson in
213
214     self.rnActor.rnSystem = TheSystem
215     and self.rnActor = TheActor
216
217     /* PostF01 */
218     AMessage.eq('Cactus Bascharage is a huge supermarket situated in the south of Luxembourg.')
219     and TheActor.rnInterfaceIN^ieMessage(AMessage)
220    }
221
222    /* Post Protocol:*/
223    postP {
224     true
225    }
226   }
227  }
228 }
```

Listing A.7: Messir Spec. file actPerson.msr.

A.8 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/ctAdministrator.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.concepts.primarytypes.classes.ctAdministrator
      {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
15 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
16
17  Operation Model {
18
19   operation: ctAdministrator.init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean {
20
21    /* Post Functional: */
22    postF {
23     if (
24      let Self:ctAdministrator in
25
```

```
26      /* Post F01 */
27       Self.login(Alogin)
28       and Self.pwd = Apwd
29       and Self.vpIsLogged = false
30
31       /* Post F02 */
32       and (Self.oclIsNew and self = Self)
33      )
34      then (result = true)
35      else (result = false)
36       endif
37     }
38   }
39  }
40 }
```

Listing A.8: Messir Spec. file ctAdministrator.msr.

A.9 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/ctCoordinator.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations.environment.primarytypes.classes.ctCoordinator
      {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
15 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
16
17  Operation Model {
18
19   operation: ctCoordinator.init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword, AaccessRights:
       etCrisisType):ptBoolean {
20
21    /* Post Functional: */
22    postF {
23     if (
24      let Self:ctCoordinator in
25
26      /* Post F01 */
27      Self.id = Aid
28      and Self.login = Alogin
29      and Self.pwd = Apwd
30      and Self.vpIsLogged = false
31      and Self.accessRights = AaccessRights
32
33      /* Post F02 */
34      and (Self.oclIsNew and self = Self)
35     )
36     then (result = true)
37     else (result = false)
38     endif
39    }
40   }
41  }
42 }
```

Listing A.9: Messir Spec. file ctCoordinator.msr.

A.10 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/ctPerson.msr

```
/*
* @author mikel
* @date Fri Mar 16 11:46:38 CET 2018
*/

package lu.uni.lassy.excalibur.MyCrash.G02.operations.concepts.primarytypes.classes.ctPerson {

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

import lu.uni.lassy.excalibur.MyCrash.G02.environment
import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes

 Operation Model {

  operation: ctPerson.init(Aid:dtPhoneNumber, Alogin:dtLogin, Apwd:dtPassword, Atype:etPersonType):
      ptBoolean {

   postF {
    if (
     let Self:ctPerson in

     /* Post F01 */

     Self.id = Aid
     and Self.login = Alogin
     and Self.pwd = Apwd
     and Self.type = Atype
     and Self.phoneNumber = AphoneNumber

      /* Post F02 */
      and (Self.oclIsNew and self = Self)
     )
     then (result = true)
     else (result = false)
      endif
    }
  }
 }
}
```

Listing A.10: Messir Spec. file ctPerson.msr.

A.11 File ./src-gen/messir-spec/environment/environment.msr

```
/*
* @author mikel
* @date Fri Mar 16 11:46:38 CET 2018
*/

package lu.uni.lassy.excalibur.MyCrash.G02.environment {

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

import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
import lu.uni.lassy.excalibur.MyCrash.G02.concepts.secondarytypes.datatypes

 Environment Model {
```

```
17
18  actor actMsrCreator role rnactMsrCreator cardinality[1 .. 1] {
19
20   operation init():ptBoolean
21
22   input interface inactMsrCreator {
23
24    operation ieSystemCreated():ptBoolean
25   }
26
27   output interface outactMsrCreator {
28
29    operation oeCreateSystemAndEnvironment():ptBoolean
30   }
31  }
32
33  actor actAuthenticated role rnactAuthenticated cardinality[1 .. *] {
34
35   operation init():ptBoolean
36
37   input interface inactAuthenticated {
38
39    operation ieLogin(AMessage:dtMessage):ptBoolean
40    operation ieLoggedOut(AMessage:dtMessage):ptBoolean
41    operation ieMessage(AMessage:dtMessage):ptBoolean
42    operation ieResetPassword():ptBoolean
43   }
44
45   output interface outactAuthenticated {
46
47    operation oeLogin(ALogin:dtLogin, APassword:dtPassword):ptBoolean
48    operation oeLogout():ptBoolean
49    operation oeLoginWithCaptcha(ALogin:dtLogin, APassword:dtPassword, ACaptcha:dtCaptcha):ptBoolean
50    operation oeResetPassword(ALogin:dtLogin):ptBoolean
51   }
52  }
53
54  actor actAdministrator role rnactAdministrator cardinality[1 .. 1] extends actAuthenticated {
55
56   operation init():ptBoolean
57
58   input interface inactAdministrator {
59
60    //PI variant
61    operation ieRequestList():ptBoolean
62    operation ieRequestBeingTreated():ptBoolean
63    operation ieRequestSolved():ptBoolean
64
65    operation iePIAdded():ptBoolean
66    operation iePIUpToDate():ptBoolean
67    operation iePIDeleted():ptBoolean
68
69    //Access rights variant
70    operation ieCoordinatorAdded():ptBoolean
71    operation ieCoordinatorDeleted():ptBoolean
72    operation ieCoordinatorUpdated():ptBoolean
73   }
74
75   output interface outactAdministrator {
76
77    // PI variant
78    operation oeGetAllRequestsFromCoordinator():ptBoolean
79    operation oeTreatRequest(ARequestID:dtID):ptBoolean
80    operation oeSolveRequest(ARequestID:dtID):ptBoolean
81
82    operation oeAddPI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:dtGPSLocation,
          APIDescription:dtDescription, APICategory:etCategory):ptBoolean
83    operation oeUpdatePI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:dtGPSLocation,
          APIDescription:dtDescription, APICategory:etCategory):ptBoolean
84    operation oeDeletePI(APIID:dtID):ptBoolean
```

```
85
86    //Access rights
87    operation oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin, AdtPassword:
          dtPassword, CoordinatorAccessRights:etCrisisType):ptBoolean
88    operation oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID):ptBoolean
89    operation oeUpdateCoordinatorAccessRights(AdtCoordinatorID:dtCoordinatorID,
          CoordinatorAccessRights:etCrisisType):ptBoolean
90
91   }
92  }
93
94  actor actCoordinator role rnactCoordinator cardinality[1 .. *] extends actAuthenticated {
95
96   operation init():ptBoolean
97
98   input interface inactCoordinator {
99
100   //PI variant
101   operation ieRequestListToCheck():ptBoolean
102   operation ieRequestDelivered():ptBoolean
103
104   //Access rights variant
105   operation ieCrisisSet():ptBoolean
106   operation ieAlertSet():ptBoolean
107  }
108
109  output interface outactCoordinator {
110
111   //PI variant
112   operation oeGetAllRequests():ptBoolean
113   operation oeCheckAvailability(ARequestID:dtID):ptBoolean
114   operation oeDeliverRequest(ARequestID:dtID):ptBoolean
115
116   //Access rights variant
117   operation oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean
118   operation oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean
119   operation oeGetAlertSet(AetAlertStatus:etAlertStatus):ptBoolean
120
121   operation oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean
122   operation oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean
123   operation oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean
124   operation oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoolean
125   operation oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisStatus):ptBoolean
126   operation oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType):ptBoolean
127  }
128 }
129
130 actor actPerson role rnactPerson cardinality[1 .. *] extends actAuthenticated {
131
132  operation init():ptBoolean
133
134  input interface inactPerson {
135
136   //PI variant
137   operation iePIAdded():ptBoolean
138  }
139
140  output interface outactPerson {
141
142   //PI variant
143   operation oeSearchPI(APIName:dtName, APICategory:etCategory, APICity:dtCity):ptBoolean
144   operation oeSendNewRequest(APIID:dtID, APIName:dtName, APICategory:etCategory, APICity:dtCity):
          ptBoolean
145
146   operation oeGetGPSLocation(APIID:dtID):ptBoolean
147   operation oeGetDescription(APIID:dtID):ptBoolean
148  }
149 }
150
151 actor actComCompany role rnactComCompany cardinality[1 .. *] {
```

```
152    input interface inactComCompany {
153
154     //Access rights variant
155     operation ieSmsSend(APIName:dtName, APICategory:etCategory, ACity:dtCity, AdtPhoneNumber:
            dtPhoneNumber, AdtSMS:dtSMS):ptBoolean
156    }
157    output interface outactComCompany {
158
159     //Access rights variant
160     operation oeAlert(APersonType:etPersonType, APIName:dtName, APIGPSLocation:dtGPSLocation,
            AdtDate:dtDate, AdtTime:dtTime, AProblemDescription:dtDescription, PhoneNumber:dtPhoneNumber
            ):ptBoolean
161    }
162   }
163
164  actor actActivator role rnactActivator cardinality[1 .. 1] {
165    input interface inactActivator {
166    }
167    output interface outactActivator {
168
169     operation oeSollicitateCrisisHandling():ptBoolean
170     operation oeSetClock(AcurrentTime:dtDateAndTime):ptBoolean
171    }
172   }
173  }
174 }
```

Listing A.11: Messir Spec. file environment.msr.

A.12 File ./src-gen/messir-spec/operations/operations.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.operations {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Operation Model {
14
15  }
16 }
```

Listing A.12: Messir Spec. file operations.msr.

A.13 File ./src-gen/messir-spec/concepts/primarytypes-associations/primarytypes-associations.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.associations {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.environment
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes
```

```
15
16 Concept Model {
17
18  Primary Types {
19
20   association assctAlertctCrisis
21    ctAlert(rnAlerts)[1..*]
22    ctCrisis(rnTheCrisis)[1..1]
23
24   association assctAlertctPerson
25    ctAlert(rnSignaled)[0..*]
26     ctPerson(rnSignaler)[1..1]
27
28    association assctCoordinatorctAdministrator
29    ctCoordinator(rnCreated)[0..*]
30    ctAdministrator(rnCreator)[1..1]
31
32    association assctAlertctCoordinator
33    ctAlert(rnValidated)[0..*]
34    ctCoordinator(rnValidator)[0..1]
35
36    association assctCrisisctCoordinator
37    ctCrisis(rnHandled)[0..*]
38    ctCoordinator(rnHandler)[0..1]
39
40   association assctRequestctCoordinator
41    ctRequest(rnChecked)[0..*]
42    ctCoordinator(rnChecker)[0..1]
43
44   association assctRequestctAdministrator
45    ctRequest(rnBeingTreated)[0..*]
46    ctAdministrator(rnTreater)[0..1]
47
48   association assctPIctAdministrator
49    ctPI(rnManaged)[0..*]
50    ctAdministrator(rnManager)[0..1]
51
52   association assctPIctPerson
53    ctPI(rnRequested)[0..*]
54    ctPerson(rnRequester)[1..1]
55
56   // Actors' associations
57
58   association assctPersonactComCompany
59     ctPerson(rnctHuman)[0..*]
60     actComCompany(rnactComCompany)[1..1]
61
62    association assctPersonactPerson
63     ctPerson(rnctPerson)[1..1]
64     actPerson(rnactPerson)[1..1]
65
66    association assctAuthenticatedactAuthenticated
67     ctAuthenticated(rnctAuthenticated)[1..1]
68     actAuthenticated(rnactAuthenticated)[1..1]
69
70    association assctCoordinatoractCoordinator
71     ctCoordinator(rnctCoordinator)[1..1]
72     actCoordinator(rnactCoordinator)[1..1]
73
74    association assctAdministratoractAdministrator
75     ctAdministrator(rnctAdministrator)[1..1]
76     actAdministrator(rnactAdministrator)[1..1]
77  }
78 }
79 }
```

Listing A.13: Messir Spec. file primarytypes-associations.msr.

A.14 File ./src-gen/messir-spec/concepts/primarytypes-classes/primarytypes-classes.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.classes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12 import lu.uni.lassy.messir.libraries.primitives
13
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
15
16  Concept Model {
17
18   Primary Types {
19
20    state class ctState {
21
22     attribute nextValueForAlertID:dtInteger
23     attribute nextValueForCrisisID:dtInteger
24     attribute clock:dtDateAndTime
25     attribute crisisReminderPeriod:dtSecond
26     attribute maxCrisisReminderPeriod:dtSecond
27     attribute vpLastReminder:dtDateAndTime
28     attribute vpStarted:ptBoolean
29
30     operation init(AnextValueForAlertID:dtInteger, AnextValueForCrisisID:dtInteger, Aclock:
           dtDateAndTime, AcrisisReminderPeriod:dtSecond, AmaxCrisisReminderPeriod:dtSecond,
           AvpLastReminder:dtDateAndTime, AvpStarted:ptBoolean):ptBoolean
31    }
32
33    class ctAuthenticated role rnctAuthenticated cardinality[0 .. *] {
34
35     attribute login:dtLogin
36     attribute pwd:dtPassword
37     attribute vpIsLogged:ptBoolean
38     attribute tries:dtInteger
39     attribute lastAccess:dtInteger
40     attribute capReq:ptBoolean
41     attribute cap2Solve:dtCaptcha
42
43     operation init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean
44
45    }
46
47    class ctAdministrator role rnctAdministrator cardinality[1 .. 1] extends ctAuthenticated {
48
49     operation init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean
50    }
51
52    class ctCoordinator role rnctCoordinator cardinality[0 .. *] extends ctAuthenticated {
53
54     attribute id:dtCoordinatorID
55     attribute accessRights:etCrisisType
56
57     operation init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword, Aaccessrights:etCrisisType)
           :ptBoolean
58    }
59
60    class ctPerson role rnctPerson cardinality[0 .. *] extends ctAuthenticated {
61
62     attribute id:dtPhoneNumber
63     attribute personType:etPersonType
```

```
64
65      operation init(Aid:dtPhoneNumber, Alogin:dtLogin, Apwd:dtPassword, ApersonType:etPersonType):
            ptBoolean
66    }
67
68    class ctRequest role rnctRequest cardinality[0 .. *] {
69
70     attribute id:dtID
71     attribute name:dtName
72     attribute city:dtCity
73     attribute category:etCategory
74     attribute status:etRequestStatus
75     attribute ignored:dtIgnored
76
77     operation init(Aid:dtID, Aname:dtName, Acity:dtCity, Acategory:etCategory, Astatus:
           etRequestStatus, Aignored:dtIgnored):ptBoolean
78    }
79
80    class ctPI role rnctPI cardinality[0 .. *] {
81
82     attribute id:dtID
83     attribute name:dtName
84     attribute city:dtCity
85     attribute category:etCategory
86     attribute location:dtGPSLocation
87     attribute description:dtDescription
88
89     operation init(Aid:dtID, Aname:dtName, Acity:dtCity, Acategory:etCategory, Alocation:
           dtGPSLocation, Adescription:dtDescription):ptBoolean
90
91    }
92
93    class ctAlert role rnctAlert cardinality[0 .. *] {
94
95     attribute id:dtAlertID
96     attribute status:etAlertStatus
97     attribute location:dtGPSLocation
98     attribute instant:dtDateAndTime
99     attribute comment:dtComment
100
101    operation init(Aid:dtAlertID, Astatus:etAlertStatus, Alocation:dtGPSLocation, Ainstant:
           dtDateAndTime, Acomment:dtComment):ptBoolean
102    operation isSentToCoordinator(AactCoordinator:ctCoordinator):ptBoolean
103   }
104
105   class ctCrisis role rnctCrisis cardinality[0 .. *] {
106
107    attribute id:dtCrisisID
108    attribute type:etCrisisType
109    attribute status:etCrisisStatus
110    attribute location:dtGPSLocation
111    attribute instant:dtDateAndTime
112    attribute comment:dtComment
113
114    operation init(Aid:dtCrisisID, Atype:etCrisisType, Astatus:etCrisisStatus, Alocation:
           dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean
115   }
116  }
117 }
118 }
```

Listing A.14: Messir Spec. file primarytypes-classes.msr.

A.15 File ./src-gen/messir-spec/concepts/primarytypes-datatypes/primarytypes-datatypes.msr

```
1 /*
2 * @author mikel
```

```
 3 * @date Fri Mar 16 11:46:38 CET 2018
 4 */
 5
 6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes {
 7
 8 import lu.uni.lassy.messir.libraries.calendar
 9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Concept Model {
14
15   Primary Types {
16
17    datatype dtID extends dtString {
18
19     operation is():ptBoolean
20    }
21
22    datatype dtAdministratorID extends dtString {
23
24     operation is():ptBoolean
25    }
26
27    datatype dtCoordinatorID extends dtString {
28
29     operation is():ptBoolean
30    }
31
32    datatype dtAlertID extends dtString {
33
34     operation is():ptBoolean
35    }
36
37    datatype dtCrisisID extends dtString {
38
39     operation is():ptBoolean
40    }
41
42    datatype dtLogin extends dtString {
43
44     operation is():ptBoolean
45    }
46
47    datatype dtPassword extends dtString {
48
49     operation is():ptBoolean
50    }
51
52    datatype dtCaptcha extends dtString {
53
54     operation is():ptBoolean
55    }
56
57    datatype dtName extends dtString {
58
59     operation is():ptBoolean
60    }
61
62    datatype dtCity extends dtString {
63
64     operation is():ptBoolean
65    }
66
67    datatype dtLatitude extends dtReal {
68
69     operation is():ptBoolean
70    }
71
72    datatype dtLongitude extends dtReal {
```

```
73
74      operation is():ptBoolean
75     }
76
77     datatype dtGPSLocation extends dtString {
78
79      attribute latitude:dtLatitude
80      attribute longitude:dtLongitude
81
82      operation is():ptBoolean
83     }
84
85     datatype dtDescription extends dtString {
86
87      operation is():ptBoolean
88     }
89
90     datatype dtComment extends dtString {
91
92      operation is():ptBoolean
93     }
94
95     datatype dtPhoneNumber extends dtString {
96
97      operation is():ptBoolean
98     }
99
100    datatype dtMessage extends dtString {
101
102     operation is():ptBoolean
103    }
104
105    datatype dtIgnored {
106     attribute attValue:ptBoolean
107     operation is():ptBoolean
108    }
109
110    enum etCategory {
111
112     constants["supermarket", "market", "hobby", "petrolstation", "building", "university", "school",
            "hospital"]
113     operation is():ptBoolean
114    }
115
116    enum etPersonType {
117
118     constants["witness", "victim", "anonymous"]
119     operation is():ptBoolean
120    }
121
122    enum etRequestStatus {
123
124     constants["pending", "treated", "solved"]
125     operation is():ptBoolean
126    }
127
128    enum etAlertStatus {
129
130     constants["pending", "valid", "invalid"]
131     operation is():ptBoolean
132    }
133
134    enum etCrisisType {
135
136     constants["low", "medium", "high"]
137     operation is():ptBoolean
138    }
139
140    enum etCrisisStatus {
141     constants["pending", "handled", "solved", "closed"]
```

```
142     operation is():ptBoolean
143    }
144   }
145  }
146 }
```

Listing A.15: Messir Spec. file primarytypes-datatypes.msr.

A.16 File ./src-gen/messir-spec/concepts/secondarytypes-associations/secondarytypes-associations.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.secondarytypes.associations {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Secondary Types {
16
17  }
18 }
19 }
```

Listing A.16: Messir Spec. file secondarytypes-associations.msr.

A.17 File ./src-gen/messir-spec/concepts/secondarytypes-classes/secondarytypes-classes.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.secondarytypes.classes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Concept Model {
14
15   Secondary Types {
16
17   }
18  }
19 }
```

Listing A.17: Messir Spec. file secondarytypes-classes.msr.

A.18 File ./src-gen/messir-spec/concepts/secondarytypes-datatypes/secondarytypes-datatypes.msr

```
1 /*
2 * @author mikel
```

```
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.concepts.secondarytypes.datatypes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Concept Model {
14
15   Secondary Types {
16
17    datatype dtSMS {
18
19     attribute value:ptString
20     operation is():ptBoolean
21    }
22   }
23  }
24 }
```

Listing A.18: Messir Spec. file secondarytypes-datatypes.msr.

A.19 File ./src-gen/messir-spec/tests/tests.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.tests {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Test Model {
14
15  }
16
17 }
```

Listing A.19: Messir Spec. file tests.msr.

A.20 File ./src-gen/messir-spec/usecases/usecaseinstance-oeThirdLoginWrong-uci.msr

```
1 package usecases.uci {
2  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3  import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5  Use Case Model {
6
7  }
8 }
```

Listing A.20: Messir Spec. file usecaseinstance-oeThirdLoginWrong-uci.msr.

A.21 File ./src-gen/messir-spec/usecases/usecaseinstance-suAddNewPI-ucisuAddNewPI.msr

```
1 package usecases.ucisuAddNewPI {
2
3  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
4  import lu.uni.lassy.excalibur.MyCrash.G02.environment
5
6  Use Case Model {
7
8   use case instance ucisuAddNewPI : suAddNewPI{
9
10   actors {
11
12    theCreator:actMsrCreator
13    patrick:actPerson
14    bob:actAdministrator
15    carl:actCoordinator
16   }
17
18   use case steps {
19
20    theCreator
21    executed instanceof subfunction oeCreateSystemAndEnvironment() {
22
23     ieSystemCreated() returned to theCreator
24    }
25
26    patrick
27    executed instanceof subfunction oeLogin("014092450D", "6543210") {
28
29     ieLoggedIn("You are logged in, 014092450D") returned to patrick
30    }
31
32    patrick
33    executed instanceof subfunction oeSearchPI("Cactus", "Hobby", "Esch/Alzette") {
34
35     ieMessage("The point of interest is NOT in the system!") returned to patrick
36    }
37
38    patrick
39    executed instanceof subfunction oeSendNewRequest("Cactus", "Hobby", "Esch/Alzette") {
40
41     ieMessage("We will handle your request.") returned to patrick
42    }
43
44    carl
45    executed instanceof subfunction oeLogin("coordin", "coordinChecks") {
46
47     ieLoggedIn("You are logged in, coordin") returned to carl
48    }
49
50    carl
51    executed instanceof subfunction oeGetAllRequests() {
52
53     ieRequestListToCheck() returned to carl
54    }
55
56    carl
57    executed instanceof subfunction oeCheckAvailability("1001") {
58
59     ieMessage("The PI is NOT available!") returned to carl
60    }
61
62    carl
63    executed instanceof subfunction oeDeliverRequest("1001") {
64
65     ieRequestDelivered() returned to carl
66    }
67
68    carl
69    executed instanceof subfunction oeLogout() {
70
```

```
71      ieLoggedOut("You are logged out!") returned to carl
72     }
73
74     bob
75     executed instanceof subfunction oeLogin("admin", "adminHandles") {
76
77      ieLoggedIn("You are logged in, admin") returned to bob
78     }
79
80     bob
81     executed instanceof subfunction oeGetAllRequestsFromCoordinator() {
82
83      ieRequestList() returned to bob
84     }
85
86     bob
87     executed instanceof subfunction oeTreatRequest("1001") {
88
89      ieRequestBeingTreated() returned to bob
90     }
91
92     bob
93     executed instanceof subfunction oeSolveRequest("1001") {
94
95      ieRequestSolved() returned to bob
96      ieRequestSolved() returned to patrick
97
98     }
99     bob
100    executed instanceof subfunction oeAddPI("Cactus", "Esch/Alzette", "78.8888/89.6565", "Cactus
           Hobby is a market for garden activities.", "Hobby") {
101
102     iePIAdded returned to bob
103     iePIAdded returned to patrick
104    }
105
106    bob
107    executed instanceof subfunction oeLogout() {
108
109     ieLoggedOut("You are logged out!") returned to bob
110    }
111   }
112  }
113 }
114}
```

Listing A.21: Messir Spec. file usecaseinstance-suAddNewPI-ucisuAddNewPI.msr.

A.22 File ./src-gen/messir-spec/usecases/usecaseinstance-suGenerateNewAlert-ucisuGenerateNewAlert.msr

```
1package usecaseinstances.uci {
2 import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3 import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5 Use Case Model {
6
7  use case instance ucisuGenerateNewAlert : suGenerateNewAlert {
8
9   actors {
10
11   theCreator:actMsrCreator
12   patrick:actPerson
13   orange:actComCompany
14   }
15
16   use case steps {
17
```

```
18     theCreator
19     executed instanceof subfunction oeCreateSystemAndEnvironment() {
20
21      ieSystemCreated() returned to theCreator
22     }
23
24     patrick
25     executed instanceof subfunction oeLogin("014092450D", "6543210") {
26
27      ieLoggedIn("You are logged in, 014092450D") returned to patrick
28     }
29
30     patrick
31     executed instanceof subfunction oeSearchPI("Cactus", "Supermarket", "Esch/Alzette") {
32
33      ieMessage("The point of interest is in the system. How to proceed?") returned to patrick
34     }
35
36     patrick
37     executed instanceof subfunction oeGetGPSLocation("1001") {
38
39      ieMessage("The GPS location of the PI is: 56.8989/46.7878.") returned to patrick
40      ieMessage("Your Sms has been sent to your communication operator.") returned to patrick
41      ieSmsSend("Cactus", "Supermarket", "Esch/Alzette", "661777777", "Supermarket on fire!")
            returned to orange
42     }
43
44     patrick
45     executed instanceof subfunction oeLogout() {
46
47      ieLoggedOut("You are logged out!") returned to patrick
48     }
49
50     orange
51     executed instanceof subfunction oeAlert("Witness", "Cactus", "Esch/Alzette", "27/03/2018", "
           10:00", "Supermarket on fire!", "661777777", "High") {
52
53      ieMessage("Alert has been sent. It will be handled!") returned to orange
54     }
55    }
56   }
57  }
58 }
```

Listing A.22: Messir Spec. file usecaseinstance-suGenerateNewAlert-ucisuGenerateNewAlert.msr.

A.23 File ./src-gen/messir-spec/usecases/usecaseinstance-suGlobalCrisisHandling-ucisuGlobalCrisisHandling.msr

```
1 package usecases.ucisuGlobalCrisisHandling {
2  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3  import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5  Use Case Model {
6
7   use case instance ucisuGlobalCrisisHandling : suGlobalCrisisHandling{
8    actors {
9     bob:actCoordinator
10    steve:actCoordinator
11   }
12   use case steps {
13
14    bob
15    executed instanceof subfunction oeLogin("bob", "password") {
16     ieLogin("You successfully logged in!") returned to bob
17    }
18
19    steve
```

```
20    executed instanceof subfunction oeLogin("steve", "password") {
21     ieLogin("You successfully logged in!") returned to steve
22    }
23
24    bob
25    executed instanceof subfunction oeGetAlertSet("pending", "2") {
26     ieMessage("AlertSet") returned to bob
27    }
28
29    bob
30    executed instanceof subfunction oeValidateAlert("1") {
31     ieMessage("Permission denied! Your access rights are to low to validate this alert!") returned
           to bob
32    }
33
34    steve
35    executed instanceof subfunction oeGetAlertSet("pending", "2") {
36     ieMessage("AlertSet") returned to steve
37    }
38
39    steve
40    executed instanceof subfunction oeValidateAlert("1") {
41     ieMessage("Alert validated") returned to steve
42    }
43
44    bob
45    executed instanceof subfunction oeGetCrisisSet("pending", "2") {
46     ieMessage("CrisisSet") returned to bob
47    }
48
49    bob
50    executed instanceof subfunction oeSetCrisisHandler("1") {
51     ieMessage("Permission denied! Your access rights are to low to handle this crisis!") returned
           to bob
52    }
53
54    steve
55    executed instanceof subfunction oeGetCrisisSet("pending", "1") {
56     ieMessage("CrisisSet") returned to steve
57    }
58
59    steve
60    executed instanceof subfunction oeSetCrisisHandler("1") {
61     ieMessage("Crisis handler set to steve") returned to steve
62    }
63
64    steve
65    executed instanceof subfunction oeSetCrisisStatus("1", "in progression") {
66     ieMessage("Crisis status set to in progression") returned to steve
67    }
68
69    steve
70    executed instanceof subfunction oeReportOnCrisis("1", "Crisis handled") {
71     ieMessage("Report added to the crisis") returned to steve
72    }
73
74    steve
75    executed instanceof subfunction oeCloseCrisis("1") {
76     ieMessage("Crisis closed") returned to steve
77    }
78
79    steve
80    executed instanceof subfunction oeSetCrisisType("2", "1") {
81     ieMessage("Crisis type set to 1") returned to steve
82    }
83
84    bob
85    executed instanceof subfunction oeGetCrisisSet("pending", "1") {
86     ieMessage("CrisisSet") returned to bob
87    }
```

```
88
89      bob
90      executed instanceof subfunction oeSetCrisisHandler("2") {
91       ieMessage("Crisis handler set to bob") returned to bob
92      }
93
94      bob
95      executed instanceof subfunction oeSetCrisisStatus("2", "in progression") {
96       ieMessage("Crisis status set to in progression") returned to bob
97      }
98
99      bob
100     executed instanceof subfunction oeReportOnCrisis("2", "Crisis handled") {
101      ieMessage("Report added to the crisis") returned to bob
102     }
103
104     bob
105     executed instanceof subfunction oeCloseCrisis("2") {
106      ieMessage("Crisis closed") returned to bob
107     }
108
109     bob
110     executed instanceof subfunction oeLogout() {
111      ieLogout("You successfully logged out!") returned to bob
112     }
113
114     steve
115     executed instanceof subfunction oeLogout() {
116      ieLogout("You successfully logged out!") returned to steve
117     }
118    }
119   }
120  }
121 }
```

Listing A.23: Messir Spec. file usecaseinstance-suGlobalCrisisHandling-ucisuGlobalCrisisHandling.msr.

A.24 File ./src-gen/messir-spec/usecases/usecaseinstance-suScenarioPresentation-ucisuScenarioPresentation.msr

```
1 package usecases.ucisuScenarioPresentation {
2  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3  import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5  Use Case Model {
6
7   use case instance ucisuScenarioPresentation : suScenarioPresentation{
8    actors {
9     theCreator:actMsrCreator
10    patrick:actPerson
11    admin:actAdministrator
12    carl:actCoordinator
13    bob:actCoordinator
14    orange:actComCompany
15   }
16   use case steps {
17    theCreator
18    executed instanceof subfunction oeCreateSystemAndEnvironment() {
19
20     ieSystemCreated() returned to theCreator
21    }
22
23    admin
24    executed instanceof subfunction oeLogin("icrashadmin", "7WXC1359") {
25     ieLogin("You successfully logged in!") returned to admin
26    }
27
```

```
28    admin
29    executed instanceof subfunction oeAddCoordinator("1", "bob", "password", "low") {
30     ieCoordinatorAdded() returned to admin
31    }
32
33    admin
34    executed instanceof subfunction oeAddCoordinator("2", "carl", "password", "medium") {
35     ieCoordinatorAdded() returned to admin
36    }
37
38    bob
39    executed instanceof subfunction oeLogin("bob", "pass") {
40     ieLogin("Wrong password! Tries left: 2") returned to bob
41    }
42
43    bob
44    executed instanceof subfunction oeLogin("bob", "pass") {
45     ieLogin("Wrong password! Tries left: 1") returned to bob
46    }
47
48    bob
49    executed instanceof subfunction oeLogin("bob", "pass") {
50     ieLogin("Wrong password! Tries left: 0.") returned to bob
51     ieLogin("Login with captcha required!") returned to bob
52    }
53
54    bob
55    executed instanceof subfunction oeLoginWithCaptcha("bob", "password", "267324") {
56     ieLogin("You successfully logged in!") returned to bob
57    }
58
59    patrick
60    executed instanceof subfunction oeLogin("014092450D", "6543210") {
61
62     ieLoggedIn("You are logged in, 014092450D") returned to patrick
63    }
64
65    patrick
66    executed instanceof subfunction oeSearchPI("Cactus", "Hobby", "Esch/Alzette") {
67
68     ieMessage("The point of interest is NOT in the system!") returned to patrick
69    }
70
71    patrick
72    executed instanceof subfunction oeSendNewRequest("Cactus", "Hobby", "Esch/Alzette") {
73
74     ieMessage("We will handle your request.") returned to patrick
75    }
76
77    carl
78    executed instanceof subfunction oeLogin("carl", "password") {
79
80     ieLoggedIn("You are logged in, carl") returned to carl
81    }
82
83    carl
84    executed instanceof subfunction oeGetAllRequests() {
85
86     ieRequestListToCheck() returned to carl
87    }
88
89    carl
90    executed instanceof subfunction oeCheckAvailability("1001") {
91
92     ieMessage("The PI is NOT available!") returned to carl
93    }
94
95    carl
96    executed instanceof subfunction oeDeliverRequest("1001") {
97
```

```
98      ieRequestDelivered() returned to carl
99     }
100
101    admin
102    executed instanceof subfunction oeGetAllRequestsFromCoordinator() {
103
104     ieRequestList() returned to admin
105    }
106
107    admin
108    executed instanceof subfunction oeTreatRequest("1001") {
109
110     ieRequestBeingTreated() returned to admin
111    }
112
113    admin
114    executed instanceof subfunction oeSolveRequest("1001") {
115
116     ieRequestSolved() returned to admin
117     ieRequestSolved() returned to patrick
118
119    }
120
121    admin
122    executed instanceof subfunction oeAddPI("Cactus", "Esch/Alzette", "78.8888/89.6565", "Cactus
           Hobby is a market for garden activities.", "Hobby") {
123
124     iePIAdded returned to admin
125     iePIAdded returned to patrick
126    }
127
128    admin
129    executed instanceof subfunction oeLogout() {
130
131     ieLoggedOut("You are logged out!") returned to admin
132    }
133
134    patrick
135    executed instanceof subfunction oeSearchPI("Cactus", "Supermarket", "Esch/Alzette") {
136
137     ieMessage("The point of interest is in the system. How to proceed?") returned to patrick
138    }
139
140    patrick
141    executed instanceof subfunction oeGetGPSLocation("Cactus", "Supermarket", "Esch/Alzette") {
142
143     ieMessage("The GPS location of the PI is: 56.8989/46.7878.") returned to patrick
144     ieMessage("Your Sms has been sent to your communication operator.") returned to patrick
145     ieSmsSend("Cactus", "Supermarket", "Esch/Alzette", "661777777", "Supermarket on fire!")
            returned to orange
146    }
147
148    patrick
149    executed instanceof subfunction oeLogout() {
150
151     ieLoggedOut("You are logged out!") returned to patrick
152    }
153
154    orange
155    executed instanceof subfunction oeAlert("Witness", "Cactus", "Esch/Alzette", "27/03/2018", "
           10:00", "Supermarket on fire!", "661777777", "Medium") {
156
157     ieMessage("Alert has been sent. It will be handled!") returned to orange
158    }
159
160    bob
161    executed instanceof subfunction oeGetAlertSet("pending") {
162     ieMessage("AlertList") returned to bob
163    }
164
```

```
165     bob
166     executed instanceof subfunction oeValidateAlert("1") {
167      ieMessage("Alert validated") returned to bob
168     }
169
170     bob
171     executed instanceof subfunction oeGetCrisisSet("pending", "medium") {
172      ieMessage("Crisis List") returned to bob
173     }
174
175     bob
176     executed instanceof subfunction oeSetCrisisHandler("1") {
177      ieMessage("Permission denied! Your access rights are to low to handle this crisis!") returned
             to bob
178     }
179
180     carl
181     executed instanceof subfunction oeGetCrisisSet("pending", "medium") {
182      ieMessage("Crisis List") returned to carl
183     }
184
185     carl
186     executed instanceof subfunction oeSetCrisisHandler("1") {
187      ieMessage("Crisis handler set to carl") returned to carl
188     }
189
190     carl
191     executed instanceof subfunction oeSetCrisisStatus("1", "in progression") {
192      ieMessage("Crisis status set to in progression") returned to carl
193     }
194
195     carl
196     executed instanceof subfunction oeReportOnCrisis("1", "Crisis handled") {
197      ieMessage("Report added to the crisis") returned to carl
198     }
199
200     carl
201     executed instanceof subfunction oeCloseCrisis("1") {
202      ieMessage("Crisis closed") returned to carl
203     }
204
205     bob
206     executed instanceof subfunction oeLogout() {
207      ieLogout("You successfully logged out!") returned to bob
208     }
209
210     carl
211     executed instanceof subfunction oeLogout() {
212      ieLogout("You successfully logged out!") returned to carl
213     }
214    }
215   }
216  }
217 }
```

Listing A.24: Messir Spec. file usecaseinstance-suScenarioPresentation-ucisuScenarioPresentation.msr.

A.25 File ./src-gen/messir-spec/usecases/usecaseinstance-ugAdministrateTheSystem-uciugAdministrateTheSystem.msr

```
1 package usecases.uciugAdministrateTheSystem {
2  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3  import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5  Use Case Model {
6
7   use case instance uciugAdministrateTheSystem : ugAdministrateTheSystem{
8    actors {
```

```
9     admin:actAdministrator
10   }
11   use case steps {

13    admin
14    executed instanceof subfunction oeLogin("icrashadmin", "7WXC1359") {
15     ieLogin("You successfully logged in!") returned to admin
16    }

18    admin
19    executed instanceof subfunction oeAddCoordinator("1", "bob", "password", "1") {
20     ieCoordinatorAdded() returned to admin
21    }

23    admin
24    executed instanceof subfunction oeUpdateCoordinatorAccessRights("1", "2") {
25     ieCoordinatorUpdated() returned to admin
26    }

28    admin
29    executed instanceof subfunction oeDeleteCoordinator("1") {
30     ieCoordinatorDeleted() returned to admin
31    }

33    admin
34    executed instanceof subfunction oeLogout() {
35     ieLogout("You successfully logged out!") returned to admin
36    }
37   }
38  }
39 }
40 }
```

Listing A.25: Messir Spec. file usecaseinstance-ugAdministrateTheSystem-uciugAdministrateTheSystem.msr.

A.26 File ./src-gen/messir-spec/usecases/usecaseinstance-ugLoginWithCaptcha-uciugLoginWithCaptcha.msr

```
1 package usecases.uciugLoginWithCaptcha {
2  import lu.uni.lassy.excalibur.MyCrash.G02.usecases
3  import lu.uni.lassy.excalibur.MyCrash.G02.environment
4
5  Use Case Model {
6
7   use case instance uciugLoginWithCaptcha : ugLoginWithCaptcha{
8    actors {
9     bob:actAuthenticated
10   }
11   use case steps {

13    bob
14    executed instanceof subfunction oeLogin("bob", "qwertz") {
15     ieLogin("Wrong Password, try again.") returned to bob
16    }

18    bob
19    executed instanceof subfunction oeLogin("bob", "idontknowanymore") {
20     ieLogin("Wrong Password, try again.") returned to bob
21    }

23    bob
24    executed instanceof subfunction oeLogin("bob", "nothing") {
25     ieLogin("Wrong Password, try again.") returned to bob
26     ieLogin("Login with captcha required.") returned to bob
27     //ieLoginWrongSendCaptcha("C4p7cha") returned to bob
28    }
29
```

```
30     bob
31     executed instanceof subfunction oeLoginWithCaptcha("bob", "sendhelp", "C4p7cha") {
32      ieLogin("Welcome bob.") returned to bob
33     }
34    }
35   }
36  }
37 }
```

Listing A.26: Messir Spec. file usecaseinstance-ugLoginWithCaptcha-uciugLoginWithCaptcha.msr.

A.27 File ./src-gen/messir-spec/usecases/usecases.msr

```
1 /*
2 * @author mikel
3 * @date Fri Mar 16 11:46:38 CET 2018
4 */
5
6 package lu.uni.lassy.excalibur.MyCrash.G02.usecases {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.primarytypes.datatypes
14 import lu.uni.lassy.excalibur.MyCrash.G02.concepts.secondarytypes.datatypes
15
16 import lu.uni.lassy.excalibur.MyCrash.G02.environment
17
18 import lu.uni.lassy.messir.libraries.primitives
19
20  Use Case Model {
21   use case system subfunction oeCreateSystemAndEnvironment() {
22
23    actor actMsrCreator[primary, active]
24
25    returned messages {
26
27     ieSystemCreated returned to actMsrCreator
28    }
29   }
30
31   use case system subfunction oeLogin(ALogin:dtLogin, APassword:dtPassword) {
32
33    actor actAuthenticated[primary, active]
34
35    returned messages {
36
37     ieLogin returned to actAuthenticated
38    }
39   }
40
41   use case system subfunction oeLogout() {
42
43    actor actAuthenticated[primary, active]
44
45    returned messages {
46
47     ieLoggedOut returned to actAuthenticated
48    }
49   }
50
51   use case system subfunction oeGetAllRequestsFromCoordinator() {
52
53    actor actAdministrator[primary, active]
54    actor actCoordinator[secondary, passive]
55    actor actPerson[secondary, multiple]
56
```

```
57   returned messages {
58
59    ieRequestList returned to actAdministrator
60   }
61  }
62
63  use case system subfunction oeTreatRequest(ARequestID:dtID) {
64
65   actor actAdministrator[primary, active]
66   actor actPerson[secondary, passive]
67  }
68
69  use case system subfunction oeSolveRequest(ARequestID:dtID) {
70
71   actor actAdministrator[primary, active]
72   actor actPerson[secondary, passive]
73
74   returned messages {
75
76    ieRequestSolved returned to actAdministrator
77   }
78  }
79
80  use case system subfunction oeAddPI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:
        dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory) {
81
82   actor actAdministrator[primary, active]
83   actor actPerson[secondary, passive]
84
85   returned messages {
86
87    iePIAdded returned to actAdministrator
88    iePIAdded returned to actPerson
89   }
90  }
91
92  use case system subfunction oeUpdatePI(APIID:dtID, APIName:dtName, APICity:dtCity, APIGPSLocation:
        dtGPSLocation, APIDescription:dtDescription, APICategory:etCategory) {
93
94   actor actAdministrator[primary, active]
95
96   returned messages {
97
98    iePIUpToDate returned to actAdministrator
99   }
100 }
101
102 use case system subfunction oeDeletePI(APIID:dtID) {
103
104  actor actAdministrator[primary, active]
105
106  returned messages {
107
108   iePIDeleted returned to actAdministrator
109  }
110 }
111
112 use case system subfunction oeGetAllRequests() {
113
114  actor actCoordinator[primary, active]
115  actor actPerson[secondary, multiple]
116
117  returned messages {
118
119   ieRequestListToCheck returned to actCoordinator
120  }
121 }
122
123 use case system subfunction oeCheckAvailability(ARequestID:dtID) {
124
```

```
125   actor actCoordinator[primary, active]
126   actor actPerson[secondary, passive]
127
128   returned messages {
129
130    ieMessage returned to actCoordinator
131   }
132  }
133
134  use case system subfunction oeDeliverRequest(ARequestID:dtID) {
135
136   actor actCoordinator[primary, active]
137   actor actPerson[secondary, passive]
138   actor actAdministrator[secondary, passive]
139
140   returned messages {
141
142    ieRequestDelivered returned to actCoordinator
143   }
144  }
145
146  use case system subfunction oeSearchPI(APIName:dtName, APICategory:etCategory, APICity:dtCity) {
147
148   actor actPerson[primary, active]
149   actor actAdministrator[secondary, passive]
150
151   returned messages {
152
153    ieMessage returned to actPerson
154   }
155  }
156
157  use case system subfunction oeSendNewRequest(APIID:dtID, APIName:dtName, APICategory:etCategory,
         APICity:dtCity) {
158
159   actor actPerson[primary, active]
160   actor actCoordinator[secondary, passive]
161
162   returned messages {
163
164    ieMessage returned to actPerson
165   }
166  }
167
168  use case system subfunction oeGetGPSLocation(APIID:dtID) {
169
170   actor actPerson[primary, active]
171   actor actAdministrator[secondary, passive]
172
173   returned messages {
174
175    ieMessage returned to actPerson
176   }
177  }
178
179  use case system subfunction oeGetDescription(APIID:dtID) {
180
181   actor actPerson[primary, active]
182   actor actAdministrator[secondary, passive]
183
184   returned messages {
185
186    ieMessage returned to actPerson
187   }
188  }
189
190  use case system subfunction oeLoginWithCaptcha(ALogin:dtLogin, APassword:dtPassword, ACaptcha:
         dtCaptcha) {
191
192   actor actAuthenticated[primary, active]
```

```
193
194   returned messages {
195
196     ieLogin returned to actAuthenticated
197    }
198   }
199
200   use case system subfunction oeSollicitateCrisisHandling() {
201
202    actor actActivator[primary, proactive]
203
204    returned messages {
205
206     ieMessage returned to actActivator
207    }
208   }
209
210   use case system subfunction oeSetClock(AcurrentTime:dtDateAndTime) {
211
212    actor actActivator[primary, proactive]
213
214    returned messages {
215
216     ieMessage returned to actActivator
217    }
218   }
219
220   use case system usergoal ugSecurelyUseSystem() {
221
222    actor actAuthenticated[primary, active]
223
224    reuse oeLogin[1..1]
225    reuse oeLogout[1..1]
226
227    step a: actAuthenticated executes oeLogin
228    step b: actAuthenticated executes oeLogout
229
230    ordering constraint
231     "step (a) must always precede step (b)."
232   }
233
234   use case system usergoal ugCheckRequest() {
235
236    actor actCoordinator[primary, active]
237
238    reuse ugSecurelyUseSystem[1..1]
239    reuse oeGetAllRequests[1..1]
240    reuse oeCheckAvailability[1..*]
241    reuse oeDeliverRequest[1..*]
242
243    step a: actCoordinator executes ugSecurelyUseSystem
244    step b: actCoordinator executes oeGetAllRequests
245    step c: actCoordinator executes oeCheckAvailability
246    step d: actCoordinator executes oeDeliverRequest
247
248    ordering constraint
249     "step (a) must always be performed before all the other steps."
250
251    ordering constraint
252     "Subsequently, all the steps follow after step (a) is performed."
253   }
254
255   use case system usergoal ugManageRequest() {
256
257    actor actAdministrator[primary, active]
258
259    reuse ugSecurelyUseSystem[1..1]
260    reuse oeGetAllRequestsFromCoordinator[1..1]
261    reuse oeTreatRequest[1..*]
262    reuse oeSolveRequest[1..*]
```

```
263
264   step a: actAdministrator executes ugSecurelyUseSystem
265   step b: actAdministrator executes oeGetAllRequestsFromCoordinator
266   step c: actAdministrator executes oeTreatRequest
267   step d: actAdministrator executes oeSolveRequest
268
269   ordering constraint
270    "step (a) must always be performed before all the other steps."
271
272   ordering constraint
273    "Subsequently, all the steps follow after step (a) is performed."
274  }
275
276  use case system summary suAddNewPI() {
277
278   actor actPerson[primary, active]
279   actor actMsrCreator[secondary, active]
280   actor actCoordinator[secondary, proactive]
281   actor actAdministrator[secondary, proactive]
282
283   reuse oeCreateSystemAndEnvironment[1..1]
284   reuse ugSecurelyUseSystem[1..1]
285   reuse oeSearchPI[1..*]
286   reuse oeSendNewRequest[1..*]
287
288   reuse ugCheckRequest[1..*]
289   reuse ugManageRequest[1..*]
290   reuse oeAddPI[1..*]
291
292   step a: actMsrCreator executes oeCreateSystemAndEnvironment
293   step b: actPerson executes ugSecurelyUseSystem
294   step c: actPerson executes oeSearchPI
295   step d: actPerson executes oeSendNewRequest
296   step e: actCoordinator executes ugSecurelyUseSystem
297   step f: actCoordinator executes ugCheckRequest
298   step g: actCoordinator executes ugSecurelyUseSystem
299   step h: actAdministrator executes ugManageRequest
300   step i: actAdministrator executes oeAddPI
301
302   ordering constraint
303    "step (a) is the first step to perform."
304
305   ordering constraint
306    "step (b), (d) and (f) must always be performed before (c), (e) and (g)."
307
308   ordering constraint
309    "Subsequently, all the steps follow after step (a) is performed."
310  }
311
312  use case system summary suGenerateNewAlert() {
313
314   actor actPerson[primary, active, multiple]
315   actor actComCompany[primary, active, multiple]
316   actor actMsrCreator[secondary, active]
317
318   reuse oeCreateSystemAndEnvironment[1..1]
319   reuse ugSecurelyUseSystem[1..1]
320   reuse oeSearchPI[1..*]
321   reuse oeGetGPSLocation[1..*]
322   reuse oeAlert[1..*]
323
324   step a: actMsrCreator executes oeCreateSystemAndEnvironment
325   step b: actPerson executes ugSecurelyUseSystem
326   step c: actPerson executes oeSearchPI
327   step d: actPerson executes oeGetGPSLocation
328   step e: actComCompany executes oeAlert
329
330   ordering constraint
331    "step (a) and (b) must always be performed before all the other steps."
332
```

```
333   ordering constraint
334    "step (a) is the first step to perform."
335
336   ordering constraint
337    "Subsequently, all the steps follow after step (a) is performed."
338  }
339
340  use case system subfunction oeAlert(APersonType:etPersonType, APIName:dtName, APIGPSLocation:
         dtGPSLocation, AdtDate:lu.uni.lassy.messir.libraries.calendar.dtDate, AdtTime:lu.uni.lassy.
         messir.libraries.calendar.dtTime, AProblemDescription:dtDescription, PhoneNumber:dtPhoneNumber
         ) {
341
342   actor actComCompany[primary, active]
343
344   returned messages {
345    ieSmsSend returned to actComCompany
346   }
347  }
348
349  use case system subfunction oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin,
         AdtPassword:dtPassword, CoordinatorAccessRights:etCrisisType) {
350   actor actAdministrator[primary, active]
351
352   returned messages {
353    ieCoordinatorAdded returned to actAdministrator
354   }
355  }
356
357  use case system subfunction oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID) {
358   actor actAdministrator[primary, active]
359
360   returned messages {
361    ieCoordinatorDeleted returned to actAdministrator
362   }
363  }
364
365  use case system subfunction oeUpdateCoordinatorAccessRights(AdtCoordinatorID:dtCoordinatorID,
         CoordinatorAccessRights:etCrisisType) {
366   actor actAdministrator[primary, active]
367
368   returned messages {
369    ieCoordinatorUpdated returned to actAdministrator
370   }
371  }
372
373  use case system usergoal ugAdministrateTheSystem() {
374   actor actAdministrator[primary, active]
375
376   reuse ugSecurelyUseSystem[1..1]
377   reuse oeAddCoordinator[1..*]
378   reuse oeDeleteCoordinator[1..*]
379   reuse oeUpdateCoordinatorAccessRights[1..*]
380
381   step a: actAdministrator executes ugSecurelyUseSystem
382   step b: actAdministrator executes oeAddCoordinator
383   step c: actAdministrator executes oeDeleteCoordinator
384   step d: actAdministrator executes oeUpdateCoordinatorAccessRights
385
386   ordering constraint
387    "Step a must be executed before all other steps"
388
389   ordering constraint
390    "Step b must be executed before executing steps c or d"
391  }
392
393  use case system subfunction oeGetCrisisSet(AetCrisisStatus:etCrisisStatus) {
394   actor actCoordinator[primary, active]
395
396   returned messages {
397    ieCrisisSet() returned to actCoordinator
```

```
398   }
399  }
400
401  use case system subfunction oeGetAlertSet(AetAlertStatus:etAlertStatus) {
402   actor actCoordinator[primary, active]
403
404   returned messages {
405    ieAlertSet() returned to actCoordinator
406   }
407  }
408
409  use case system usergoal ugMonitor() {
410   actor actCoordinator[primary, active]
411
412   reuse ugSecurelyUseSystem[1..1]
413   reuse oeGetCrisisSet[1..*]
414   reuse oeGetAlertSet[1..*]
415
416   step a: actCoordinator executes ugSecurelyUseSystem
417   step b: actCoordinator executes oeGetCrisisSet
418   step c: actCoordinator executes oeGetAlertSet
419
420   ordering constraint
421    "Step a must be executed before all other step"
422  }
423
424  use case system subfunction oeValidateAlert(AdtAlertID:dtAlertID) {
425   actor actCoordinator[primary, active]
426
427   returned messages {
428    ieMessage() returned to actCoordinator
429   }
430  }
431
432  use case system subfunction oeInvalidateAlert(AdtAlertID:dtAlertID) {
433   actor actCoordinator[primary, active]
434
435   returned messages {
436    ieMessage() returned to actCoordinator
437   }
438  }
439
440  use case system subfunction oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:
         etCrisisStatus) {
441   actor actCoordinator[primary, active]
442
443   returned messages {
444    ieMessage() returned to actCoordinator
445   }
446  }
447
448  use case system subfunction oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType) {
449   actor actCoordinator[primary, active]
450
451   returned messages {
452    ieMessage() returned to actCoordinator
453   }
454  }
455
456  use case system subfunction oeSetCrisisHandler(AdtCrisisID:dtCrisisID) {
457   actor actCoordinator[primary, active]
458
459   returned messages {
460    ieMessage() returned to actCoordinator
461   }
462  }
463
464  use case system subfunction oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment) {
465   actor actCoordinator[primary, active]
466
```

```
467    returned messages {
468     ieMessage() returned to actCoordinator
469    }
470   }
471
472   use case system subfunction oeCloseCrisis(AdtCrisisID:dtCrisisID) {
473    actor actCoordinator[primary, active]
474
475    returned messages {
476     ieMessage() returned to actCoordinator
477    }
478   }
479
480   use case system usergoal ugManageCrisis() {
481    actor actCoordinator[primary, active]
482
483    reuse ugSecurelyUseSystem[1..1]
484    reuse oeValidateAlert[1..*]
485    reuse oeInvalidateAlert[1..*]
486    reuse oeSetCrisisStatus[1..*]
487    reuse oeSetCrisisType[1..*]
488    reuse oeSetCrisisHandler[1..*]
489    reuse oeReportOnCrisis[1..*]
490    reuse oeCloseCrisis[1..*]
491
492    step a: actCoordinator executes ugSecurelyUseSystem
493    step b: actCoordinator executes oeValidateAlert
494    step c: actCoordinator executes oeInvalidateAlert
495    step d: actCoordinator executes oeSetCrisisHandler
496    step e: actCoordinator executes oeSetCrisisStatus
497    step f: actCoordinator executes oeSetCrisisType
498    step g: actCoordinator executes oeReportOnCrisis
499    step h: actCoordinator executes oeCloseCrisis
500
501    ordering constraint
502     "Step a must be executed before executing all other steps"
503
504    ordering constraint
505     "Step b must be executed before executing step d"
506
507    ordering constraint
508     "Step d must executed before executing steps e, f, g and h"
509   }
510
511   use case system summary suGlobalCrisisHandling() {
512    actor actCoordinator[primary, active]
513
514    reuse ugSecurelyUseSystem
515    reuse ugManageCrisis
516    reuse ugMonitor
517
518    step a: actCoordinator executes ugSecurelyUseSystem
519    step b: actCoordinator executes ugManageCrisis
520    step c: actCoordinator executes ugMonitor
521
522    ordering constraint
523     "Step a must be executed before executing all other steps"
524   }
525  use case system summary suScenarioPresentation() {
526    actor actMsrCreator[primary, active]
527    actor actAdministrator[primary, active]
528    actor actCoordinator[primary, active]
529    actor actPerson[primary, active]
530    actor actComCompany[primary, active]
531
532    reuse oeCreateSystemAndEnvironment[1..1]
533    reuse oeLoginWithCaptcha[1..*]
534    reuse oeSearchPI[1..*]
535    reuse oeSendNewRequest[1..*]
536    reuse ugCheckRequest[1..*]
```

```
537    reuse ugManageRequest[1..*]
538    reuse oeSearchPI[1..*]
539    reuse oeGetGPSLocation[1..*]
540    reuse oeAlert[1..*]
541    reuse ugAdministrateTheSystem[1..*]
542    reuse ugMonitor[1..*]
543    reuse ugManageCrisis[1..*]
544
545    step a: actMsrCreator executes oeCreateSystemAndEnvironment
546    step b: actAdministrator executes ugAdministrateTheSystem
547    step c: actCoordinator executes oeLoginWithCaptcha
548    step d: actPerson executes oeSearchPI
549    step e: actPerson executes oeSendNewRequest
550    step f: actCoordinator executes ugCheckRequest
551    step g: actAdministrator executes ugManageRequest
552    step h: actPerson executes oeSearchPI
553    step i: actPerson executes oeGetGPSLocation
554    step j: actComCompany executes oeAlert
555    step k: actCoordinator executes ugMonitor
556    step l: actCoordinator executes ugManageCrisis
557
558    ordering constraint
559     "Step a must be executed before executing all other steps"
560
561    ordering constraint
562     "Step b must follow step a"
563   }
564   use case system subfunction oeResetPassword(ALogin:dtLogin) {
565    actor actAuthenticated[primary, active]
566
567    returned messages {
568
569     ieResetPassword returned to actAuthenticated
570    }
571   }
572   use case system usergoal ugLoginWithCaptcha() {
573    actor actAuthenticated[primary, active]
574
575    reuse oeLogin[3..*]
576    reuse oeLoginWithCaptcha[1..*]
577
578    step a: actAuthenticated executes oeLogin
579    step b: actAuthenticated executes oeLogin
580    step c: actAuthenticated executes oeLogin
581    step d: actAuthenticated executes oeLoginWithCaptcha
582
583    ordering constraint
584     "Step a to c must be executed before executing all other steps"
585
586    ordering constraint
587     "Step d must be the last step"
588   }
589  }
590 }
```

Listing A.27: Messir Spec. file usecases.msr.

Bibliography

[1] Guelfi, N.: Messir: A Scientific Method for the Software Engineer. to be published (2017)

[2] Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)