VCL specification of the car-crash crisis management system

Nuno Amálio, Qin Ma, Christian Glodt and Pierre Kelsen
Laboratory for Advanced Software Systems
University of Luxembourg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg

TR-LASSY-09-03

Contents

Chapter 1

Introduction

This reports presents the abstract requirements specification of the car-crash crisis management system (CCCMS), a large-scale case study [KGM09]. This specification is described using the Visual Contract Language (VCL) [AKM10, AK09].

VCL's *package* construct makes large-scale modelling possible in VCL. A VCL package is a reusable functional unit encapsulating structure and behaviour that can be used or extended by other packages. They enable large-scale modelling and *coarse-grained* problem decomposition. VCL packages enable the definition of modules that localise a solution to some concern, which can be either crosscutting or not; such modules can be described, understood and analysed in isolation and then used as a piece in multiple contexts to make larger packages addressing multiple concerns. To enable aspect-orientation, VCL provides mechanisms to compose packages representing non-orthogonal aspects, which are typically hard to represent and compose modularly.

The remainder of this report is divided into parts:

- Part I presents a UML use case model (re-factored from the one in [KGM09]) together with UML sequence diagrams that document the interaction of the actors external to the system in the context of each use case. The objective is to derive a set of system operations to be specified using VCL.
- Part II builds a set of generic packages describing solutions to concerns of CCCMS. Some these packages are aspects, others are more classical modules.
- Part III builds packages that adapt some of the generic packages to the CCCMS context.
- Part V builds a set of VCL packages representing a decomposition of the problem domain.
- Part VI joins aspect packages with problem domain packages to build domain packages that include aspects.

- Part VII builds the sub-system packages by composing all relevant packages.
- Finally, appendix A presents the Z that would be generated from the VCL diagrams that make the VCL security packages of the case study (packages of part II).

Part I

Architecture, use cases and system operations

Chapter 2

Architecture

This chapter presents the architecture of the overall CCCMS system derived from the system's requirements [KGM09].

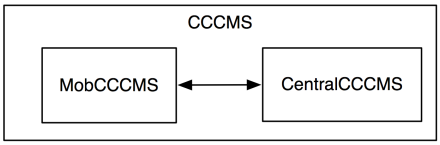


Figure 2.1: Architectural diagram highlighting the two subsystems of CCCMS.

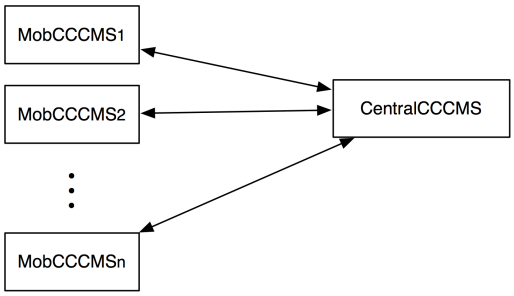


Figure 2.2: Diagram highlighting instances of the CCCMS architecture. There will be several instances of the mobile CCCMS system deployed on mobile devices that communicate with a sole central CCCMS system deployed at the Crisis Coordination headquaters.

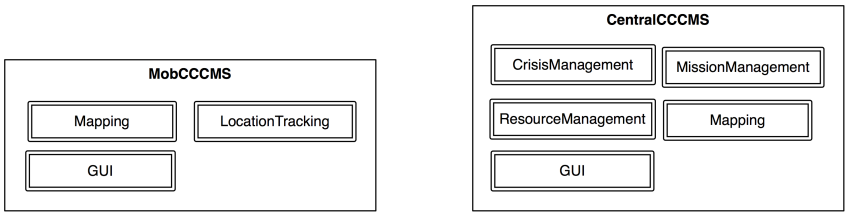


Figure 2.3: Logical architectural diagram describing logical units of functionality (double-lined boxes) in subsystems *MobCCCMS* (left) and *CentralCCCMS* (right).

Chapter 3

Use Cases

This chapter presents the UML use case model that is the basis of the VCL specification of CCCMS. The use case model presented here is re-factored from the one given in [KGM09].

3.1 Use Cases of Central CCCMS

This section presents the UML use case model of Central CCCMS. We organize the presentation of the requirement model in sub-sections according to the logical division of its functionality.

3.1.1 Crisis Management

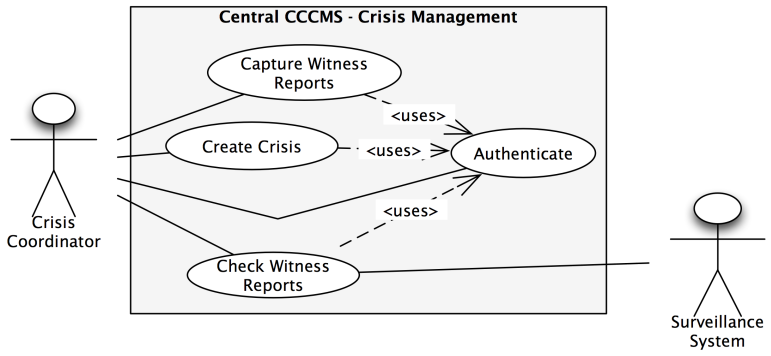


Figure 3.1: Use Case diagram documenting uses cases that are related to the *Crisis Management* logical functionality unit of the Central CCCMS.

Figure 3.1 presents the use case diagram of the crisis management functionality in the Central CCCMS. Table 3.1 presents the traceability from use cases

of [KGM09] to the ones used here.

Use Case in [KGM09]	*Use Cases used here*	*Notes*
Capture Witness Report	Capture Witness Report Check Witness Report Create Crisis	

Table 3.1: Correspondence between use cases of case study document [KGM09] and use cases that are used as basis for VCL specification.

For each use case, we document the interaction between the external actors and the system via sequence diagrams.

"Capture Witness Report" Sequence Diagram

The sequence diagram in Figure 3.2 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while recording in the system information regarding road incidents as reported by witnesses.

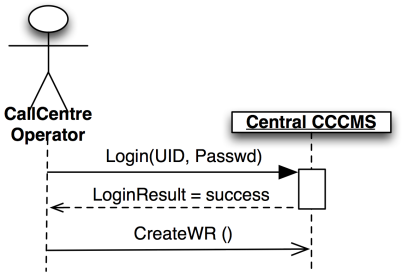


Figure 3.2: System sequence diagram documenting uses case *Capture Witness Report*.

"Check Witness Report" Sequence Diagram

The sequence diagram in Figure 3.3 documents the main success interaction scenario between the Central CCCMS system and the actors *Crisis Coordinator* and *Surveillance System*, while a crisis coordinator checks witness reports in order to take a decision regarding whether to create a crisis in the system.

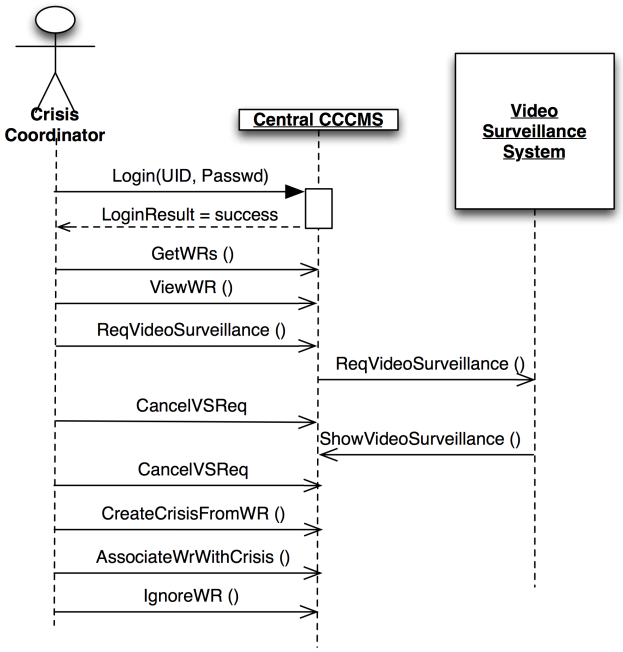


Figure 3.3: System sequence diagram documenting uses case *Check Witness Report*.

"Create Crisis" Sequence Diagram

The sequence diagram in Figure 3.4 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* for creating a crisis.

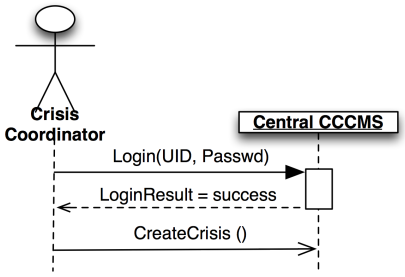


Figure 3.4: System sequence diagram documenting uses case *Create Crisis*.

3.1.2 Resource Management

Internal resources and external bodies are managed by coordinators, who can add and remove them from the system.

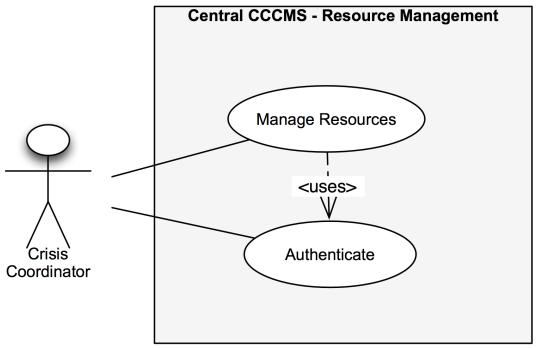


Figure 3.5: Use Case diagram documenting uses cases that are related to the *Resource Management* logical functionality unit of the Central CCCMS.

Figure 3.5 presents use case diagram for resource management in the Central CCCMS. Table 3.2 presents the traceability from use cases of [KGM09] to the ones used here.

Use Case in [KGM09]	*Use Cases used here*	*Notes*
–	Manage Resources	Introduced to cover functionality not included in [KGM09]

Table 3.2: Correspondence between use cases of case study document [KGM09] and use cases that are used as basis for VCL specification.

"Manage Resources" Sequence Diagram

The sequence diagram in Figure 3.6 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while managing internal resources or external body information in the system.

3.1.3 Mission Management

Once a crisis is created, the crisis coordinator needs to create a set of missions in order to resolve the crisis. Each mission is assigned resources, and resources may be internal or external.

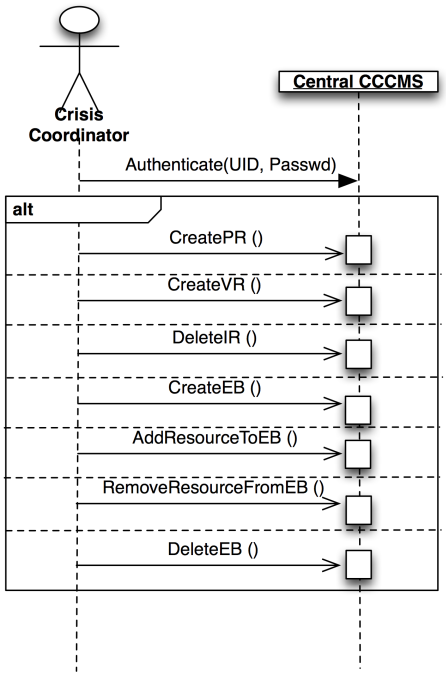


Figure 3.6: System sequence diagram documenting uses case *Manage Resources*.

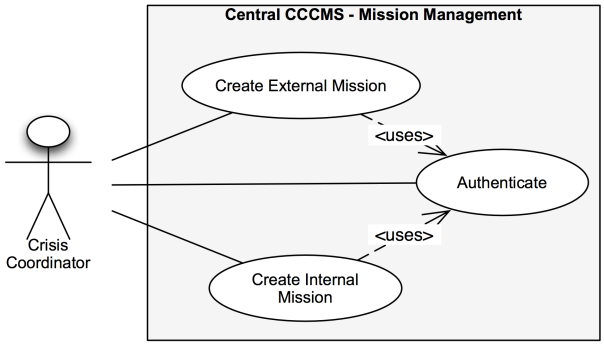


Figure 3.7: Use Case diagram documenting uses cases that are related to the *Mission Management* logical functionality unit of the Central CCCMS.

Figure 3.7 presents use case diagram for mission management in the Central CCCMS. Table 3.3 presents the traceability from use cases of [KGM09] and the ones used here.

Use Cases in [KGM09]	*Use Cases used here*	*Notes*
Assign Internal Resource Execute Mission	Create Internal Mission	
Request External Resource Execute Mission	Create External Mission	

Table 3.3: Correspondence between use cases of case study document [KGM09] and use cases that are used as basis for VCL specification.

For each use case, we document the interaction between the external actors and the system via sequence diagrams.

"Create Internal Mission" Sequence Diagram

The sequence diagram in Figure 3.8 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while creating in the system a mission to resolve a crisis. This mission is assigned to internal resources for execution.

"Create External Mission" Sequence Diagram

The sequence diagram in Figure 3.9 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while creating in the system a mission to resolve a crisis. This mission is assigned to external resources for execution.

3.1.4 Display

Crisis coordinators can require to view the information of crisis, missions and resources that are currently existing in the system.

Figure 3.10 presents use case diagram for information display in Central CCCMS. Table 3.4 presents the traceability from use cases of [KGM09] to the ones used here.

"View Crisis" Sequence Diagram

The sequence diagram in Figure 3.11 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while displaying the information relevant to a crisis in the system.

"View Mission" Sequence Diagram

The sequence diagram in Figure 3.12 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while displaying the information relevant to a mission in the system.

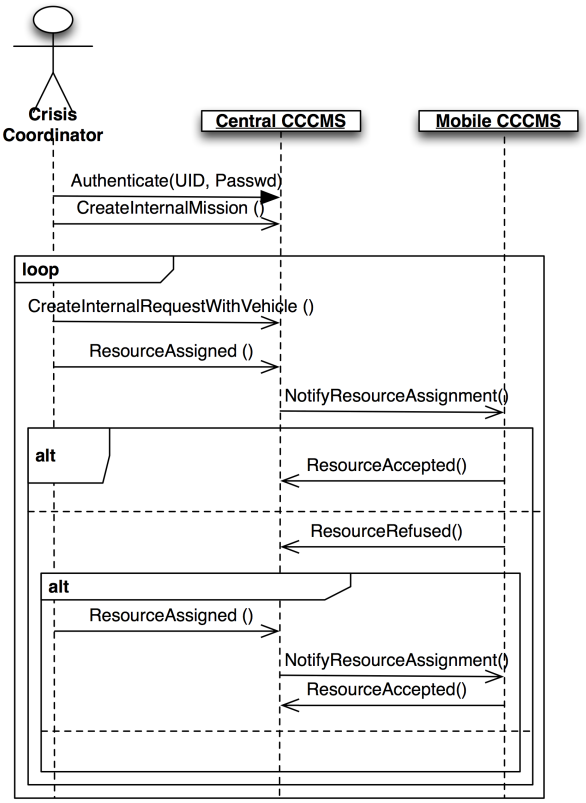


Figure 3.8: System sequence diagram documenting uses case *Create Internal Mission*.

Use Case in [KGM09]	*Use Cases used here*	*Notes*
–	View Crisis	Introduced to cover functionality not included in [KGM09]
–	View Mission	Introduced to cover functionality not included in [KGM09]
–	View Resource	Introduced to cover functionality not included in [KGM09]

Table 3.4: Correspondence between use cases of case study document [KGM09] and use cases that are used as basis for VCL specification.

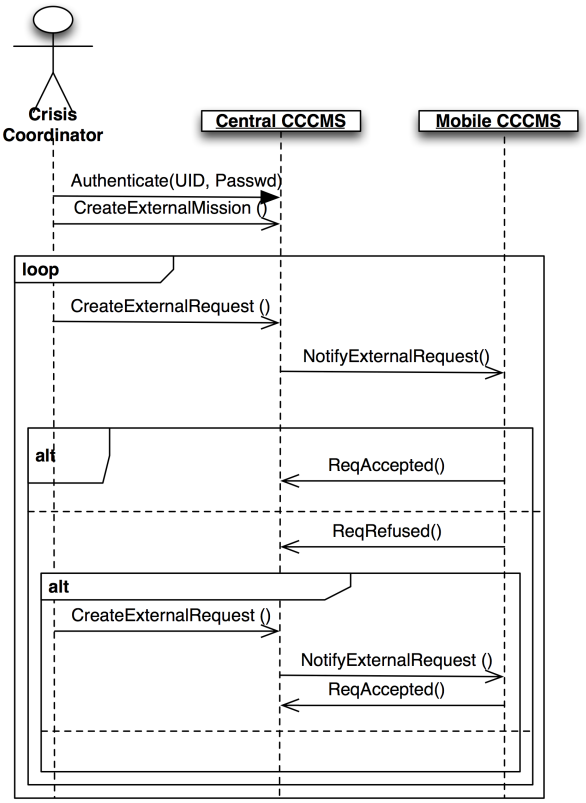


Figure 3.9: System sequence diagram documenting uses case *Create External Mission*.

"View Resource" Sequence Diagram

The sequence diagram in Figure 3.13 documents the main success interaction scenario between the Central CCCMS system and the actor *Crisis Coordinator* while displaying the information relevant to a resource in the system.

3.1.5 Mapping

3.2 Use Cases of Mobile CCCMS

The Mobile CCCMS is deployed on the mobile devices that are hold by either an internal resource agent, or an external body. Figure **??** presents use case diagram of Mobile CCCMS. The introduction of such a sub-system in CCCMS architecture comes as a result during our requirement modeling in VCL. Therefore, all the use cases presented here are new to [KGM09].

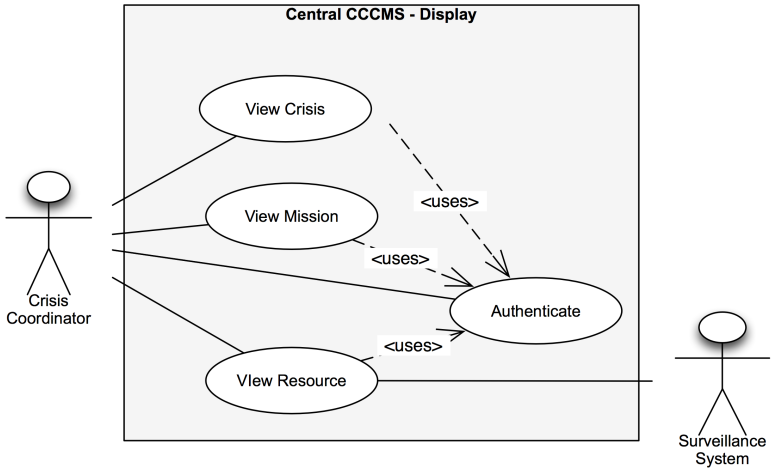


Figure 3.10: Use Case diagram documenting uses cases that are related to the *Display* logical functionality unit of the Central CCCMS.

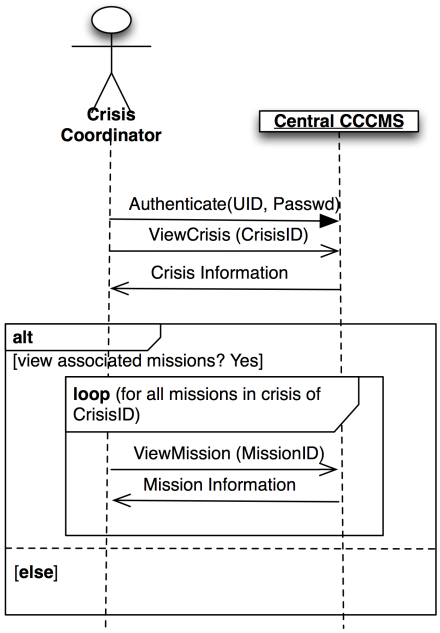


Figure 3.11: System sequence diagram documenting uses case *View Crisis*.

We present the sequence diagram of each use case in the following subsections.

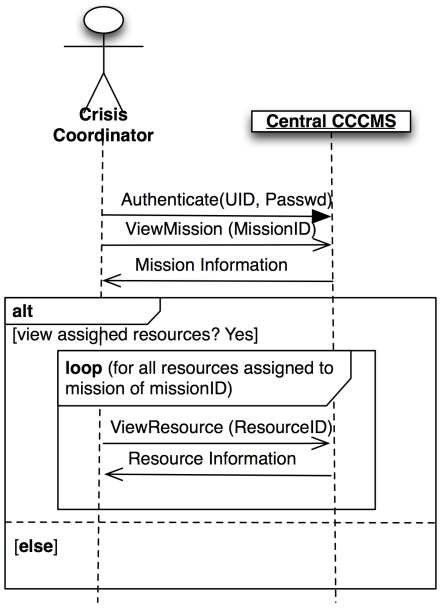


Figure 3.12: System sequence diagram documenting uses case *View Mission*.

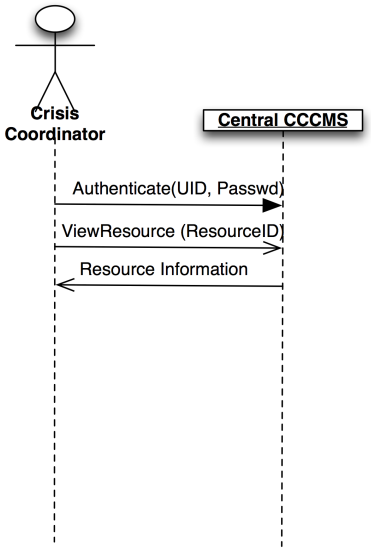


Figure 3.13: System sequence diagram documenting uses case *View Resource*.

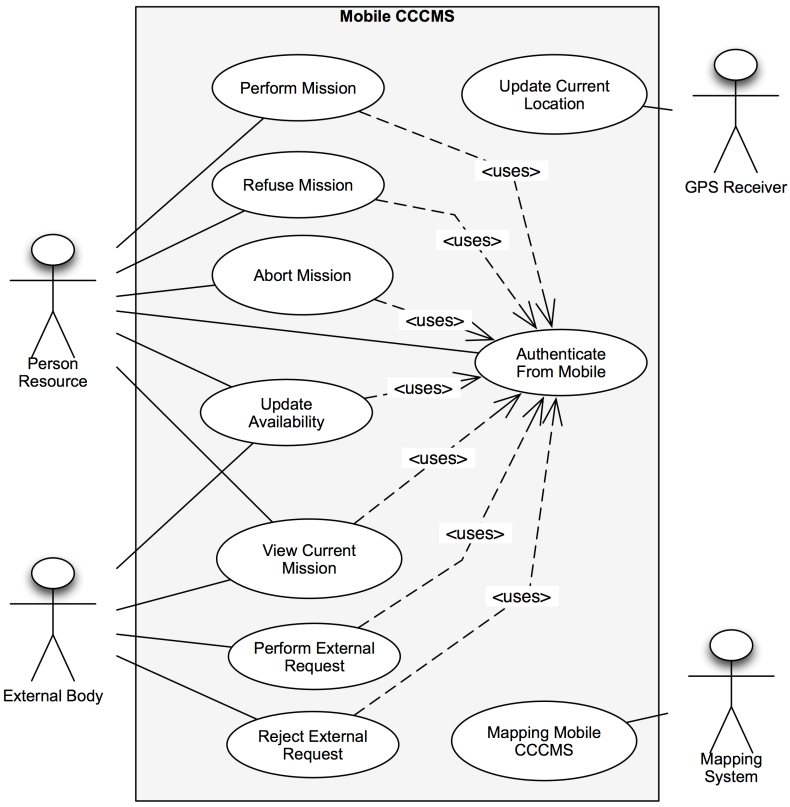


Figure 3.14: Use Case diagram documenting uses cases of the Mobile CCCMS.

3.2.1 "Perform Mission" Sequence Diagram

The sequence diagram in Figure 3.15 documents the main success interaction scenario among the actor *Person Resource*, the Mobile CCCMS system, and the Central CCCMS system while the person resource accepting and executing the assigned mission.

3.2.2 "Refuse Mission" Sequence Diagram

The sequence diagram in Figure 3.16 documents the main success interaction scenario among the actor *Person Resource*, the Mobile CCCMS system, and the Central CCCMS system while the person resource refusing the assigned mission.

3.2.3 "Abort Mission" Sequence Diagram

The sequence diagram in Figure 3.17 documents the main success interaction scenario among the actor *Person Resource*, the Mobile CCCMS system, and the

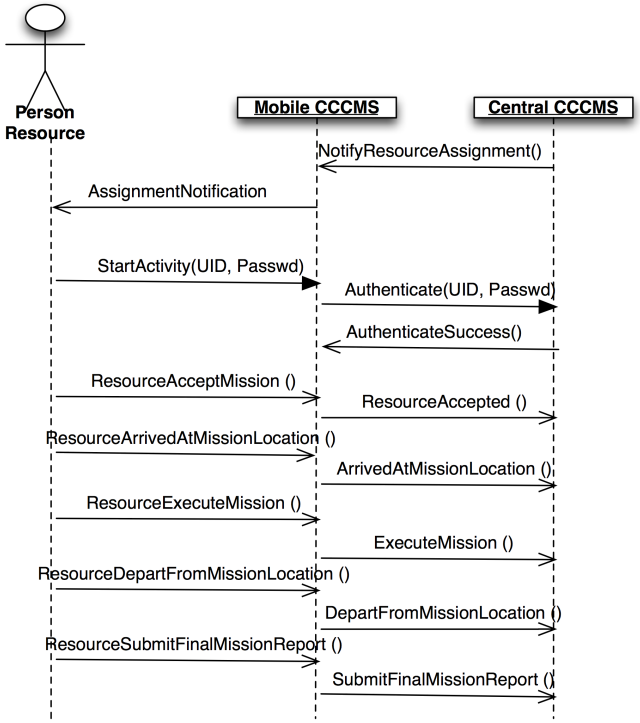


Figure 3.15: System sequence diagram documenting uses case *Perform Mission*.

Central CCCMS system while the person resource accepting but later aborting the assigned mission.

3.2.4 "Update Availability" Sequence Diagram

The resources and external bodies can each indicate their availability to the system. The sequence diagram in Figure 3.15 documents the main success interaction scenario among the actor *Person Resource* or *External Body*, the Mobile CCCMS system, and the Central CCCMS system while the person resource or the external body updating its availability status.

3.2.5 "View Current Mission" Sequence Diagram

The sequence diagram in Figure 3.19 documents the main success interaction scenario among the actor *Person Resource* or *External Body*, the Mobile CCCMS system, and the Central CCCMS system while the person resource or the external body viewing the information of the mission that is currently assigned to it.

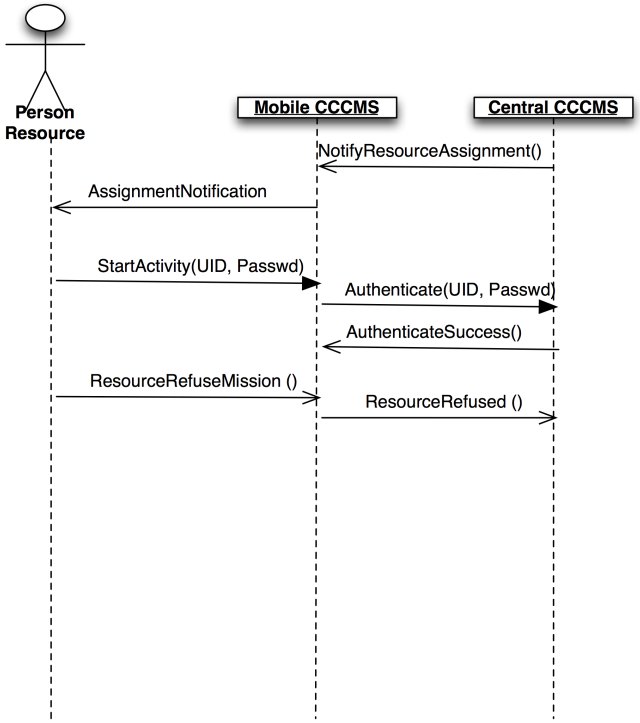


Figure 3.16: System sequence diagram documenting uses case *Refuse Mission*.

3.2.6 "Perform External Request" Sequence Diagram

The sequence diagram in Figure 3.20 documents the main success interaction scenario among the actor *External Body*, the Mobile CCCMS system, and the Central CCCMS system while the external body accepting and executing the demanded request.

3.2.7 "Reject External Request" Sequence Diagram

The sequence diagram in Figure 3.21 documents the main success interaction scenario among the actor *External Body*, the Mobile CCCMS system, and the Central CCCMS system while the external body rejecting the demanded request.

3.2.8 "Update Current Location" Sequence Diagram

3.2.9 "Mapping Mobile CCCMS" Sequence Diagram

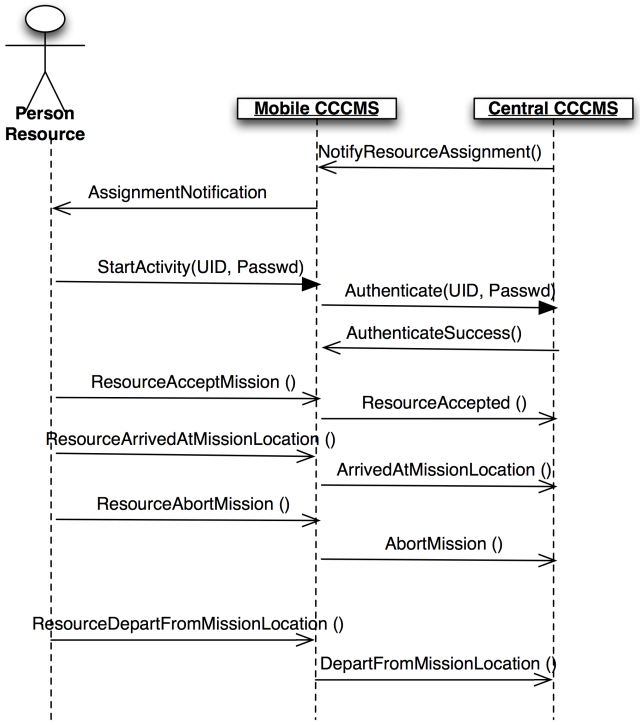


Figure 3.17: System sequence diagram documenting uses case *Abort Mission*.

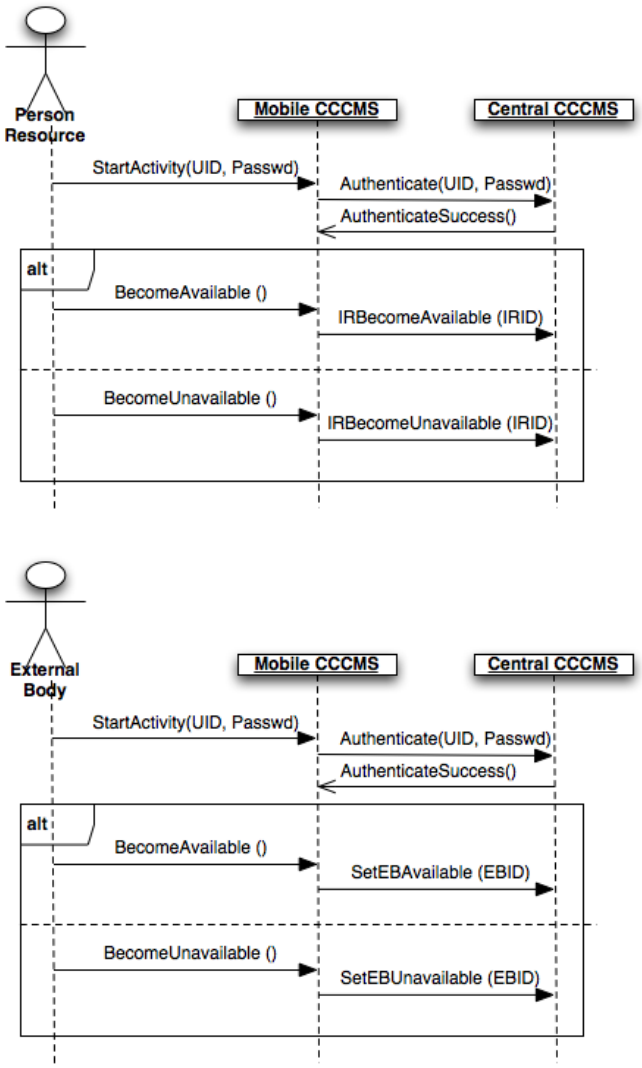


Figure 3.18: System sequence diagram documenting uses case *Update Availability*.

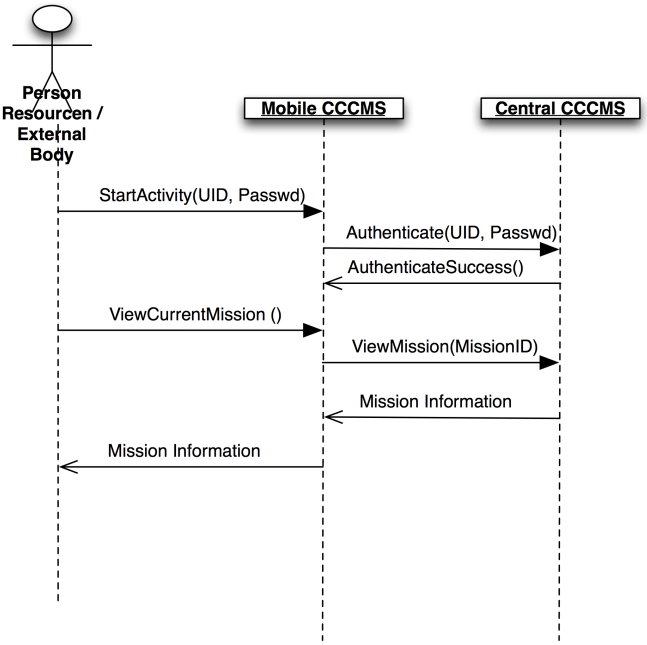


Figure 3.19: System sequence diagram documenting uses case *View Current Mission*.

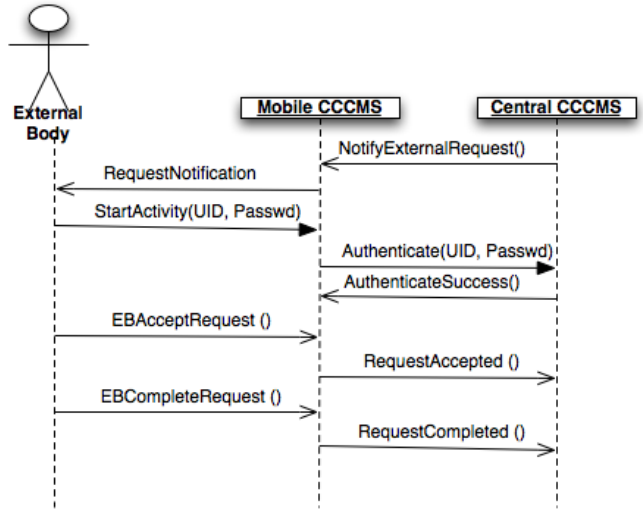


Figure 3.20: System sequence diagram documenting uses case *Perform External Request*.

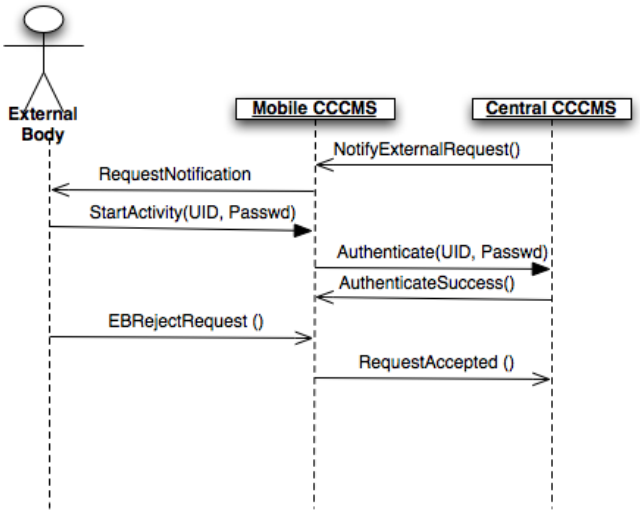


Figure 3.21: System sequence diagram documenting uses case *Reject External Request*.

Part II

Generic Packages

Chapter 4

Package *Users*

This chapter defines a package addressing the core of user-related concerns. This is to be extended by other packages defining user-related functionality.

4.1 Package Definition and Structure

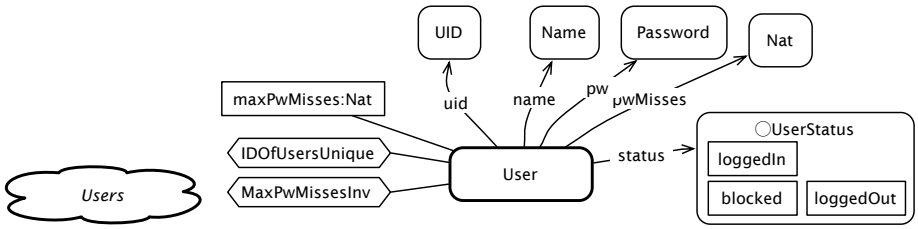


Figure 4.1: Package diagram defining package *Users* (left). Structural diagram of *Users* package (right).

Figure 4.1 (right) presents SD of package *Users*. This introduces domain blob *User*, a structure to be used in other packages dealing with user-related concerns. In diagram, value blob *UserStatus* defines a set by enumerating its elements; this says that set *UserStatus* comprises distinct elements named *loggedIn*, *loggedOut* and *blocked*. The labelled arrows emanating from *User* to other blobs define properties possessed by all elements of the set. *User* objects have a user identifier (*uid*), a user or *login* name (*uname*), the actual name of the user (*name*), a password (*pw*) and a record of the number of password misses (*pwMisses*) kept for security reasons, and a *login* status (*status*), representing the fact that a user may be *logged-out*, *logged-in* or *blocked* because number of password tries exceeded allowed maximum. Object linked to *User* defines a constant that is visible only in the scope of this blob (a local constant);

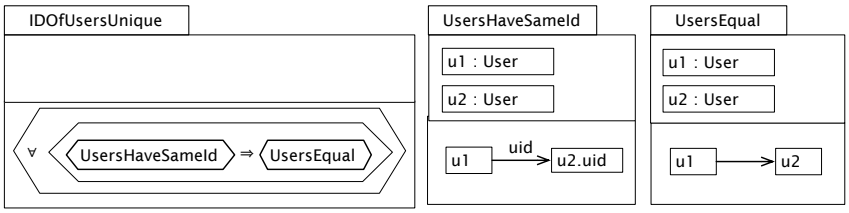


Figure 4.2: Constraint diagram for *User* 's local invariant *IDOfUsersUnique*. This says that if two users have the same id, then they must be the same user.

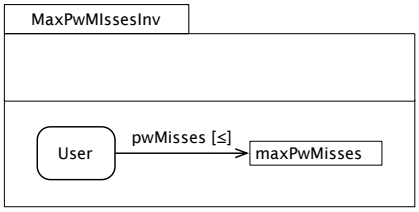


Figure 4.3: Constraint diagram for *User*'s local invariant *CntMaxPwMissesInv*.

maxPwMisses of blob *Nat* (natural numbers) represents the maximum number of allowed consecutive password misses for some user.

4.2 Behaviour

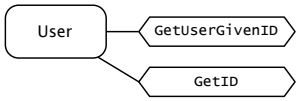


Figure 4.4: Behavioural diagram of package *Users*.

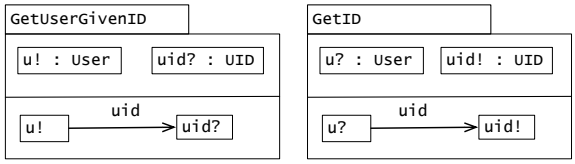


Figure 4.5: Observe Operations of blob *User GetUserGivenID* and *GetID*.

Chapter 5

Package *UsersMgmt*

This chapter defines package *UsersMgmt*, which introduces functionality for management of users (adding, deleting, modifying user information, etc). This is to be extended by other packages defining user-related functionality. Package *UsersMgmt* extends package *Users* (see chapter 4).

5.1 Package Definition

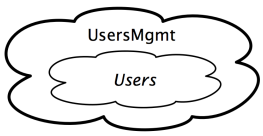


Figure 5.1: Package diagram defining package *UsersMgmt*.

5.2 Behaviour

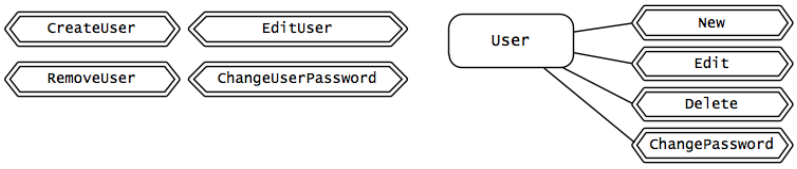


Figure 5.2: Behavioural diagram of package *UsersMgmt*.

5.2.1 Local *User* Contracts

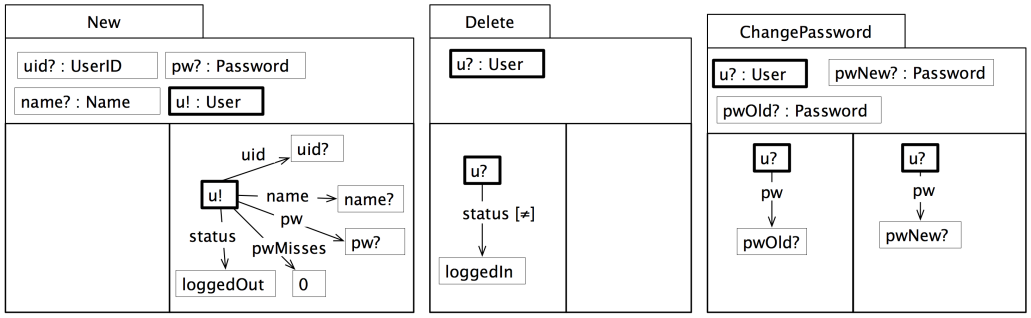


Figure 5.3: Local contract diagrams of blob *User*.

5.2.2 Global Contracts

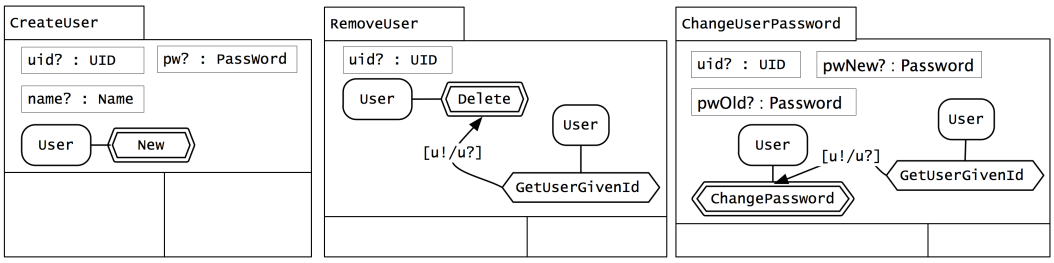


Figure 5.4: Global contracts of package *UsersMgmt*.

Chapter 6

Package *Authentication*

This chapter defines a package defining the core of a general solution to the concern of user authentication. This is to be extended by other packages to define authentication-related functionality. *Athentication* is package defining a reusable aspect that extends package *Users* (see chapter 4).

6.1 Package Definition and Structure

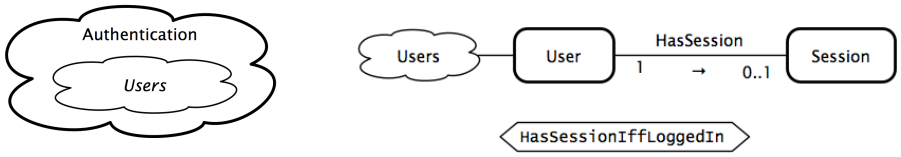


Figure 6.1: *Authentication* package extends *Users* (left). Global structural diagram of *Authentication* package (right).

Figure 6.1 presents the global structural diagram of the *Authentication* package. The diagram is as follows:

- Blob *User* represents a set of users; it comes from package *Users*.
- Blob *Session*, also a domain blob, represents a session that can be opened in the system by some user. Section 6.1.1 gives the local definition of this blob.
- Relational edge *HasSession* associate a user with its open session. A user can have at most one session open at any one time (multiplicity 0 .. 1), and a session has a user.

- Blob *LoginResult* defines a set with the possible results of a login operation: it may be successful (*loginOk*), the the login may be blocked because maximum limit for password tries has been reached (*isBlocked*), or it may just be that the password is wrong (*wrongPW*).

6.1.1 *Session* Blob

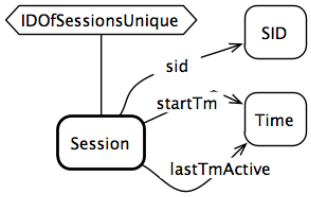


Figure 6.2: Local structural diagram of *Session* blob.

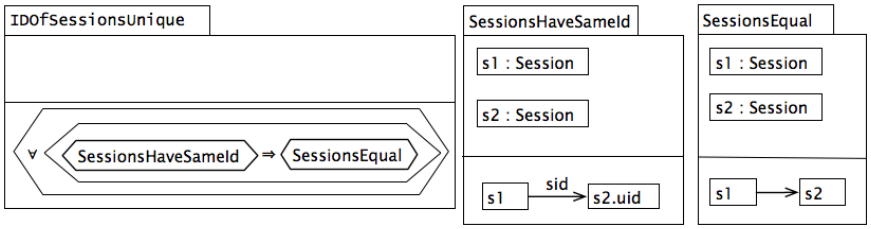


Figure 6.3: Constraint diagram for constraint *IdsOfSessionsUnique*.

6.1.2 *User* Blob

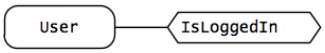


Figure 6.4: Local structural diagram of *User* blob.

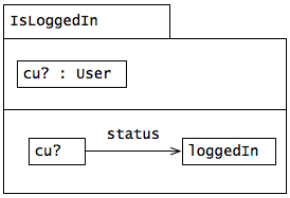


Figure 6.5: Constraint diagrams for constraint *IsLoggedIn* of Blob *User*.

6.1.3 Global constraints

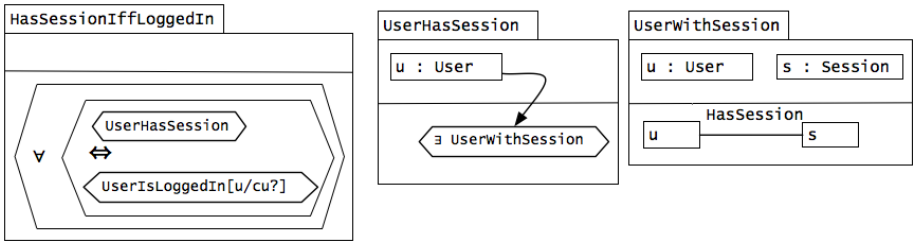


Figure 6.6: Constraint diagram for global invariant *HasSessionIfLoggedIn*.

6.2 Behaviour

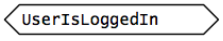


Figure 6.7: Behaviour diagram of *Authentication* package.

6.2.1 Global Behaviour

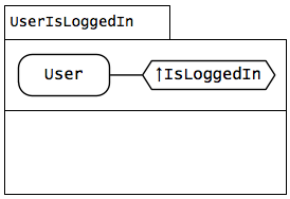


Figure 6.8: Constraint diagrams for global behavioural constraint *UserIsLoggedIn*.

Chapter 7

Package *AuthenticationOps*

This chapter defines package *AuthenticationOps* that extends package *Authentication* (see chapter 6) to provide authentication operations, such as login and logout.

7.1 Structure

Figure 7.1: *AuthenticationOps* package extends *Authentication* (left). Structural diagram of *AuthenticationOps* package (right).

7.2 Behaviour

Figure 7.2: Global behaviour diagram of *AuthenticationOps* package.

7.2.1 User

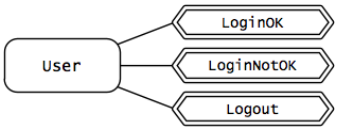


Figure 7.3: Local behaviour diagram of blob *User* in package *AuthenticationOps*.

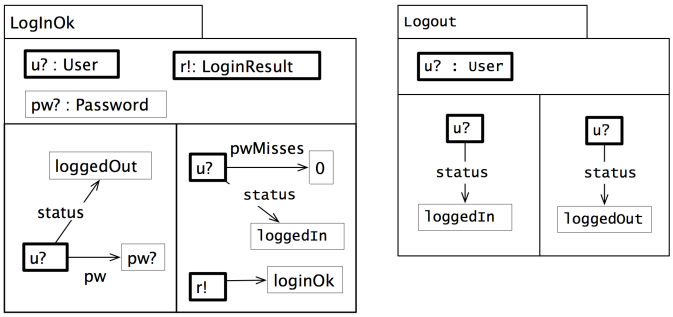


Figure 7.4: Contract Diagrams for local operations *LoginOk* and *Logout* of blob *User*.

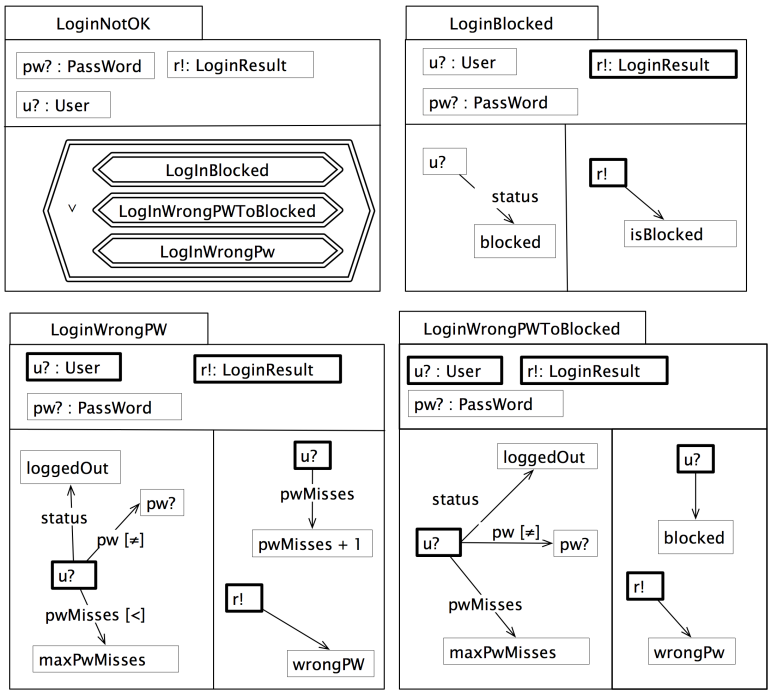


Figure 7.5: Contract Diagrams specifying local operation *LoginNotOk* of blob *User*.

7.2.2 Session

Figure 7.6: Local behaviour diagram of blob *Session*.

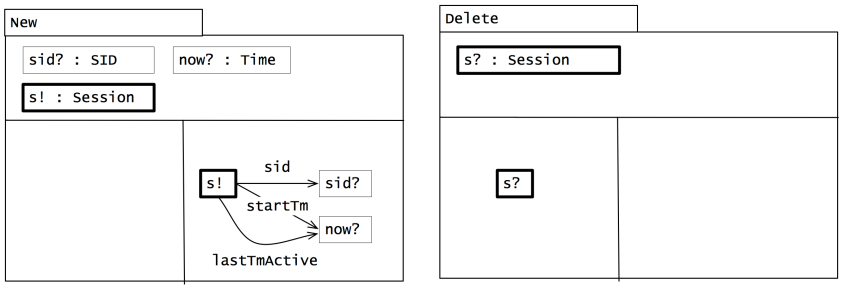


Figure 7.7: Contract Diagrams for local operations *New* and *Delete* of blob *Session*.

7.2.3 HasSession

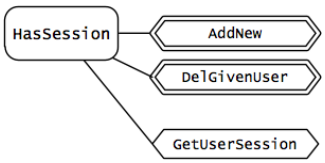


Figure 7.8: Local behaviour diagram of blob *HasSession*.

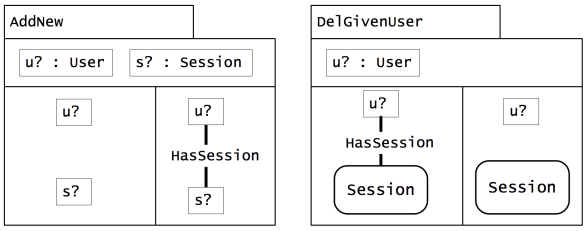


Figure 7.9: Contract Diagrams for local operations *AddUserSession* and *DelGivenUser* of relational-edge *HasSession*.

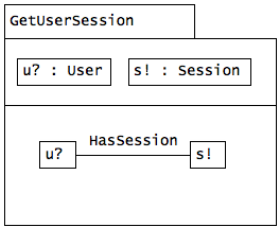


Figure 7.10: Constraint diagram of behavioural constraint *GetUserSession* of relational-edge *HasSession*.

7.2.4 Global Behaviour

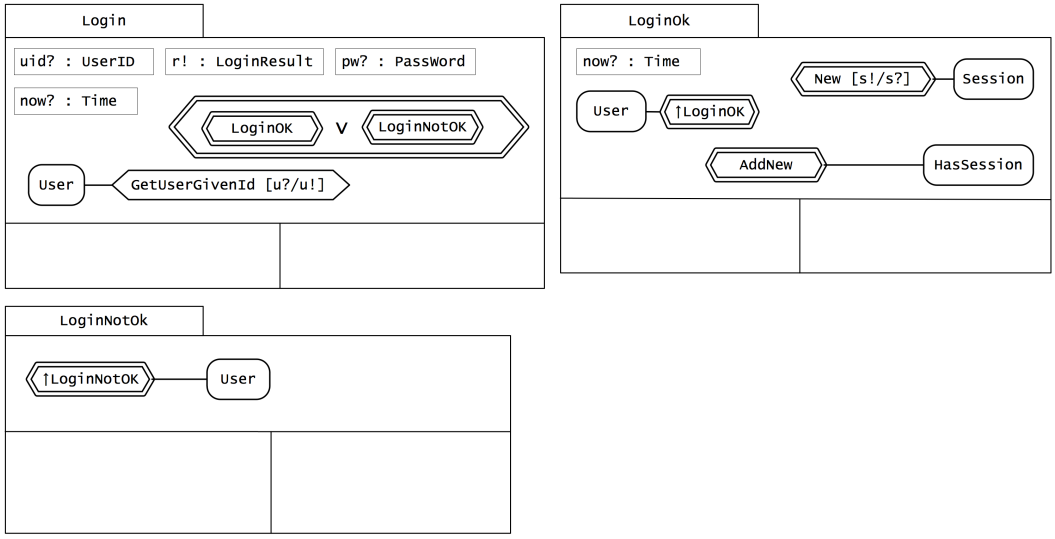


Figure 7.11: Contract Diagrams defining global operations *Login*.

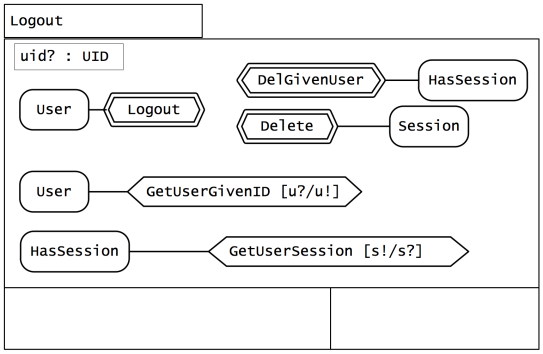


Figure 7.12: Contract Diagrams defining global operations *Logout*.

Chapter 8

Package *AuthenticationMgmt*

This chapter defines package *AuthenticationMgmt* that defines administration operations that are related with authentication, such as un-blocking and blocking users. Package *AuthenticationMgmt* extends package *Authentication* (see chapter 6).

8.1 Structure

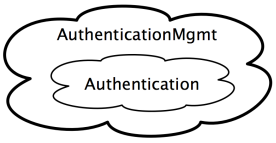


Figure 8.1: *AuthenticationMgmt* package extends *Authentication*.

8.2 Behaviour

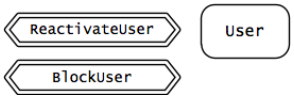


Figure 8.2: Global behavioural diagram of *AuthenticationMgmt* package.

8.2.1 User

Figure 8.3: Local behaviour diagram of blob *User* in package *AuthenticationMgmt*.

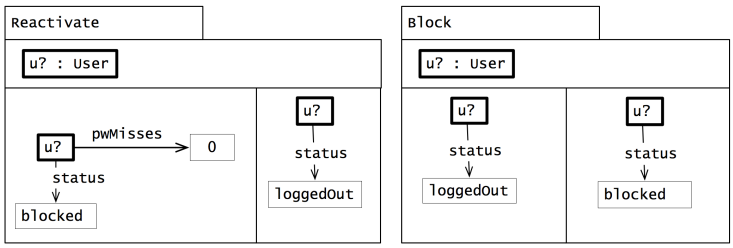


Figure 8.4: Contract diagrams of local operations *Reactivate* and *Block* of blob *User*.

8.2.2 Global Behaviour

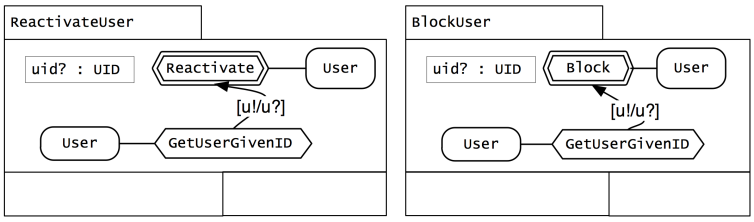


Figure 8.5: Contract diagrams of global operations *ReactivateUser* and *BlockUser*.

Chapter 9

Package *SessionMgmt*

This chapter defines package *SessionMgmt* (*Session Management*), which is concerned with managing the activity of user sessions. Package *SessionMgmt* extends package *AuthenticationOps* (see chapter 7). This package is customised to build package *SessionMgmtCCCMS* (chapter 25).

9.1 Structure

Figure 9.1: Package *SessionMgmt* extends *AuthenticationOps* (left). Structural diagram of *SessionMgmt* package (right).

9.2 Behaviour

Figure 9.2: Global behavioural diagram of *SessionMgmt* package.

9.2.1 Blob *Session*

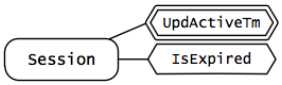


Figure 9.3: Local behavioural diagram of blob *Session* in *SessionMgmt* package.

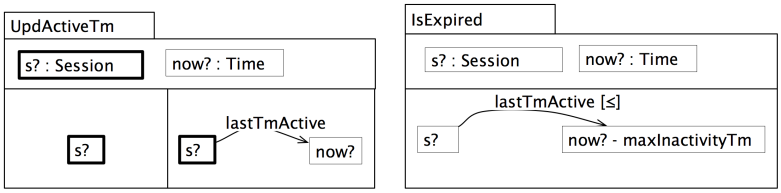


Figure 9.4: Local behavioural definitions of blob *Session*. Contract diagram of local operation *UpdActiveTm* and constraint diagram of behavioural constraint *IsExpired*.

9.2.2 Relational Edge *HasSession*

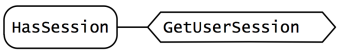


Figure 9.5: Local behavioural diagram of relational edge *HasSession* in *SessionMgmt* package.

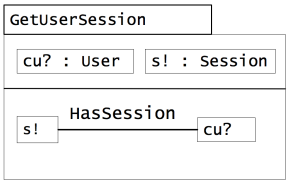


Figure 9.6: Constraint diagram of observe operation *GetUserSession*.

9.2.3 Global behaviour

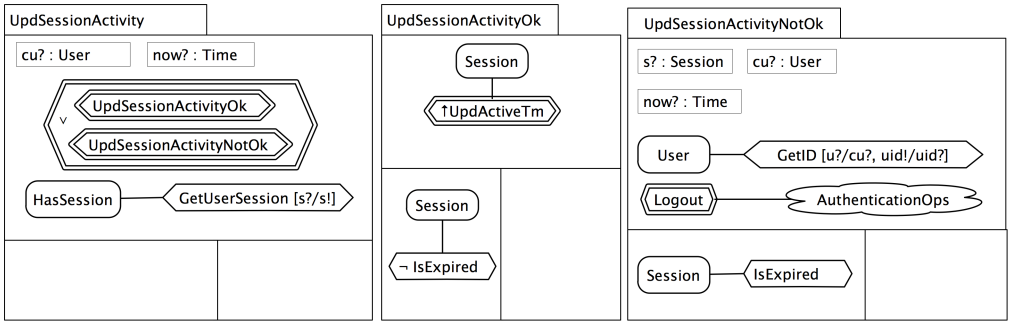


Figure 9.7: Contract diagrams of global operation *UpdSessionActivity*.

Chapter 10

Package *AccessControl*

This chapter introduces a package representing the core of a solution for rôle-based access control [SCFY96] scheme. Package *AccessControl* extends package *Users* (see chapter 4).

10.1 Structure

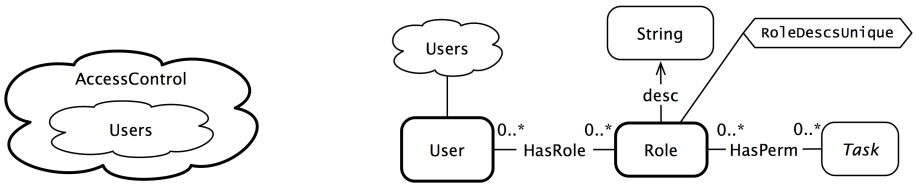
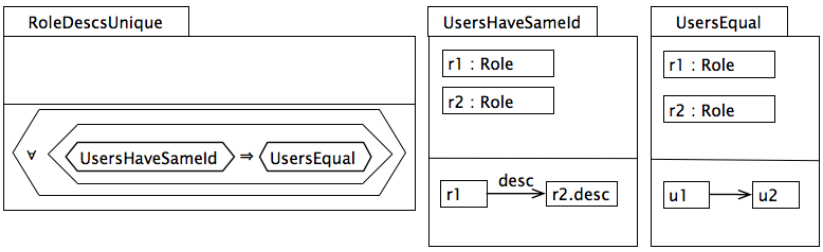


Figure 10.1: *AccessControl* package extends *Users* (left). Structural diagram of *AccessControl* package (right).

10.1.1 Blob *Role*



10.2 Behaviour

Figure 10.2: Behavioural diagram of *AccessControl* package.

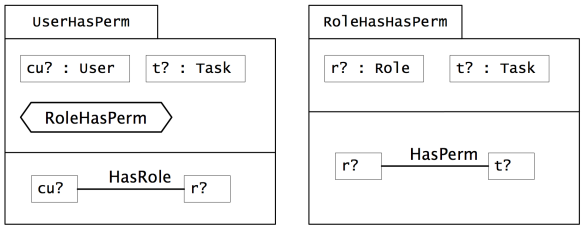


Figure 10.3: Constraint diagram defining behavioural constraint *UserHasPerm*.

Chapter 11

Package *AccessControlMgmt*

This chapter introduces package *AccessControlMgmt* (*Access-Control management*), which introduces management functionality related with access control (e.g. creating and deleting roles, role assignments and permissions). Package *AccessControlMgmt* extends package *AccessControl* (see chapter 10).

11.1 Structure

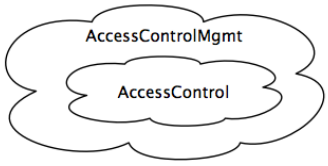


Figure 11.1: *AccessControlMgmt* package extends *AccessControl*.

11.2 Behaviour

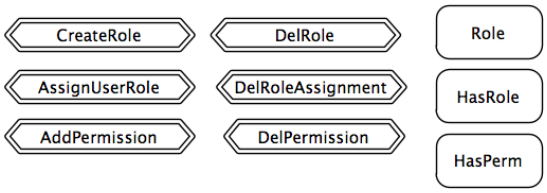


Figure 11.2: Global behavioural diagram of package *AccessControlMgmt*.

11.2.1 Blob *Role*

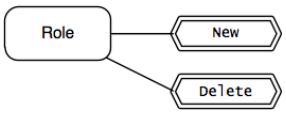


Figure 11.3: Local behaviour diagram of blob *Role* in package *AccessControlMgmt*.

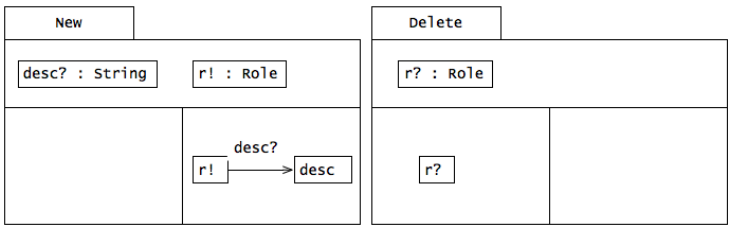


Figure 11.4: Contract diagrams of local operations *New* and *Delete* of blob *Role*.

11.2.2 Relational Edge *HasRole*

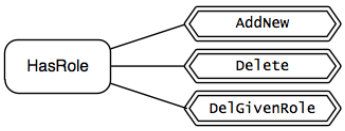


Figure 11.5: Local behaviour diagram of relational edge *HasRole*.

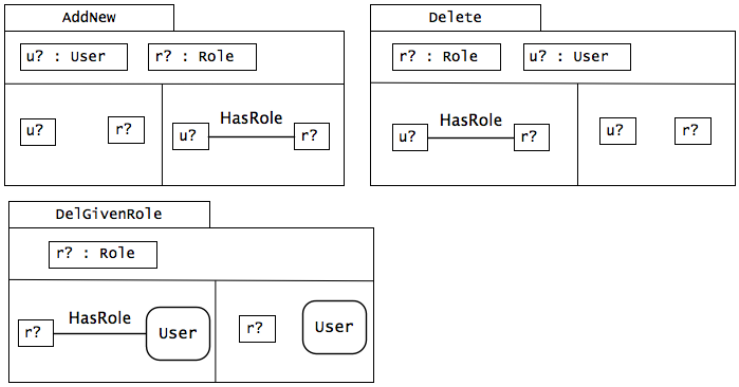


Figure 11.6: Contract diagrams defining local operations of relational edge *HasRole*.

11.2.3 Relational Edge *HasPerm*

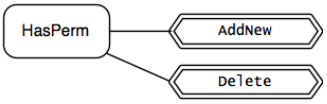


Figure 11.7: Local behaviour diagram of relational edge *HasPerm*.

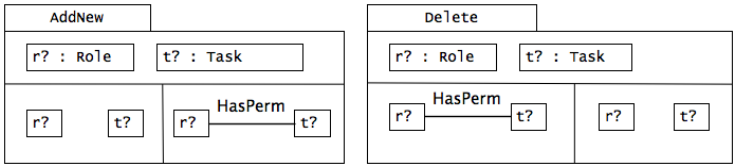


Figure 11.8: Contract diagrams defining local operations of relational edge *HasPerm*.

11.2.4 Global Behaviour

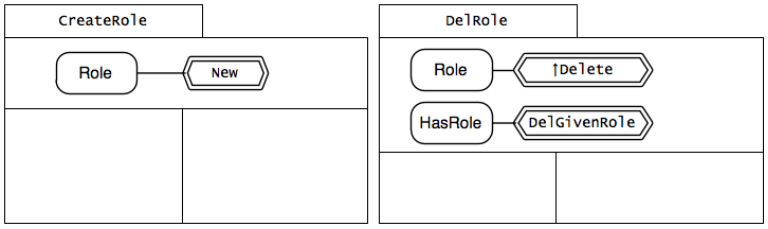


Figure 11.9: Contract diagrams defining global operations *CreateRole* and *DeleteRole*.

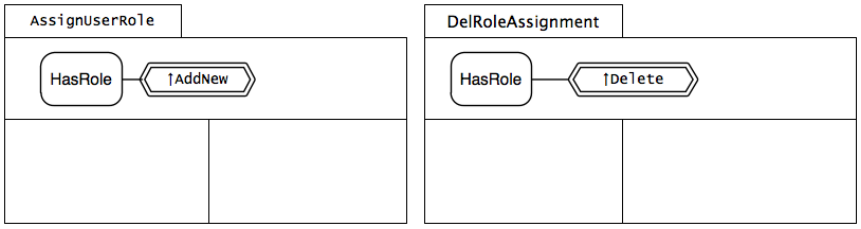


Figure 11.10: Contract diagrams defining global operations *AssignUserRole* and *DelRoleAssignment*.

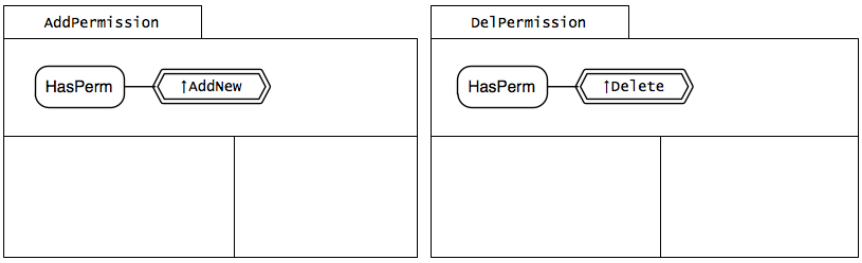


Figure 11.11: Contract diagrams defining global operations *AddPermission* and *DelPermission*.

Chapter 12

Package *Authorisation*

This chapter introduces package *Authorisation*, which provides both user authentication and rôle-based access control. Package *Authorisation* extends packages *Authentication* (chapter 6) and *AccessControl* (chapter 10).

12.1 Structure

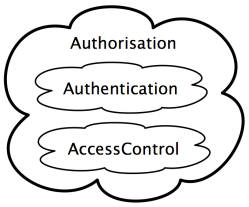


Figure 12.1: *Authorisation* package extends *Authentication* and *AccessControl*.

12.2 Behaviour

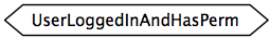


Figure 12.2: Behavioural diagram of *Authorisation* package.

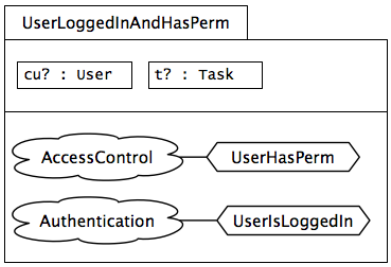


Figure 12.3: Behavioural constraint diagram *UserLoggedInAndHasPerm*.

Chapter 13

Package *SysAdmin*

This chapter introduces package *SysAdmin*, which comprises functionality related with system administration. Package *SysAdmin* extends package *Users* (chapter 4).

13.1 Structure

Figure 13.1: *SysAdmin* package extends package *Users* (left). Structural diagram of package *SysAdmin* (left).

13.2 Behaviour

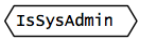


Figure 13.2: Behavioural diagram of *Authorisation* package.

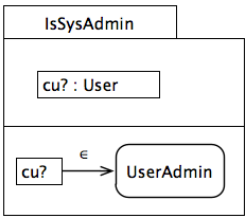


Figure 13.3: Behavioural constraint diagram *IsSysAdmin*.

Chapter 14

Package *SecAuthorisationMgmt*

This chapter introduces package *SecAuthorisationMgmt*, which provides a secure way of managing users, rôles, rôle assignments and permissions. Package *SecAuthorisationMgmt* extends packages *UsersMgmt* (chapter 5), *AuthenticationMgmt* (chapter 8), *AccessControlMgmt* (chapter 11) and *SysAdmin* (chapter 13).

14.1 Structure

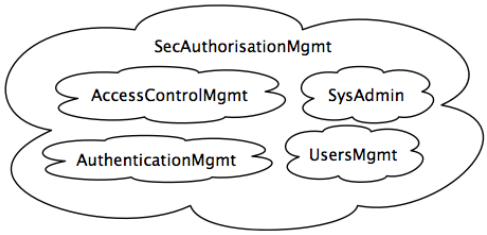


Figure 14.1: Package *SecAuthorisationMgmt* extends *AccessControlMgmt*, *AuthenticationMgmt*, *UsersMgmt* and *SysAdmin* (left).

14.2 Behaviour

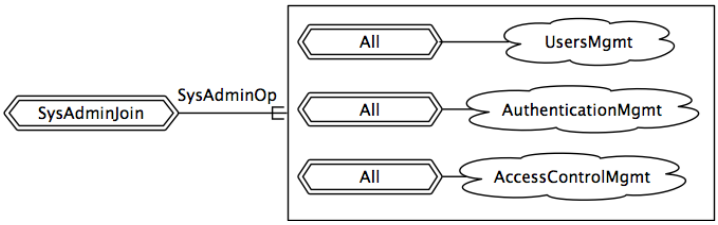


Figure 14.2: Behavioural diagram of *SecAuthorisationMgmt* package.

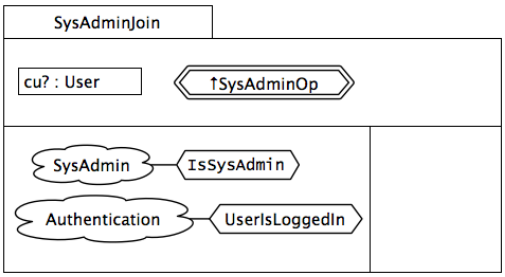


Figure 14.3: Contract diagram of join operation *SysAdminJoin*.

Chapter 15

Package *Logging*

This chapter introduces package *Logging*, which provides functionality enabling logging the activity of a system by recording event occurrences.

15.1 Structure

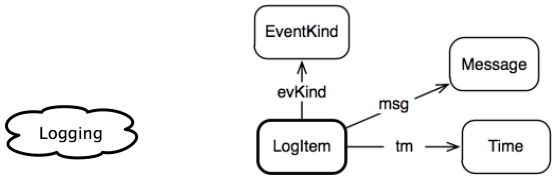


Figure 15.1: Package *Logging* (left). Structural diagram of this package (right).

15.2 Behaviour

Figure 15.2: Behavioural diagram of *Logging* package.

15.2.1 Blob *LogItem*

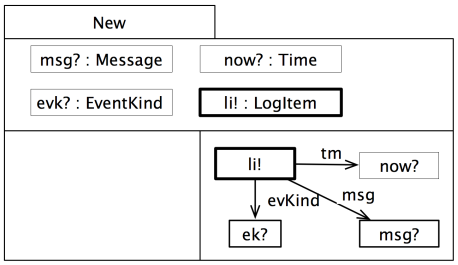


Figure 15.3: Contract diagram of *LogItem* operation *New*.

15.2.2 Global Behaviour

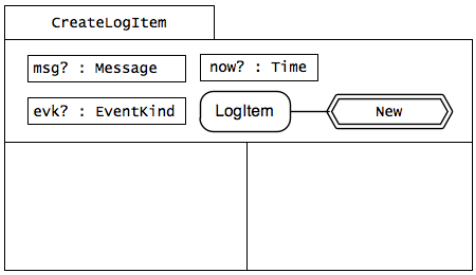


Figure 15.4: Contract diagram of global operation *CreateLogItem*.

Chapter 16

Package *MappingCommon*

This chapter introduces VCL package *MappingCommon*, which defines structures that are common across several packages that use mapping functionality.

16.1 Structure

Figure 16.1: Package *MappingCommon* (left) and its structural diagram (right).

Chapter 17

Package *Mapping*

This chapter introduces VCL package *Mapping*, which localises the generic concern related with the handling of maps. This package assumes an external mapping system, from which maps can be requested. Package *Mapping* extends package *MappingCommon* (chapter 16)

17.1 Structure

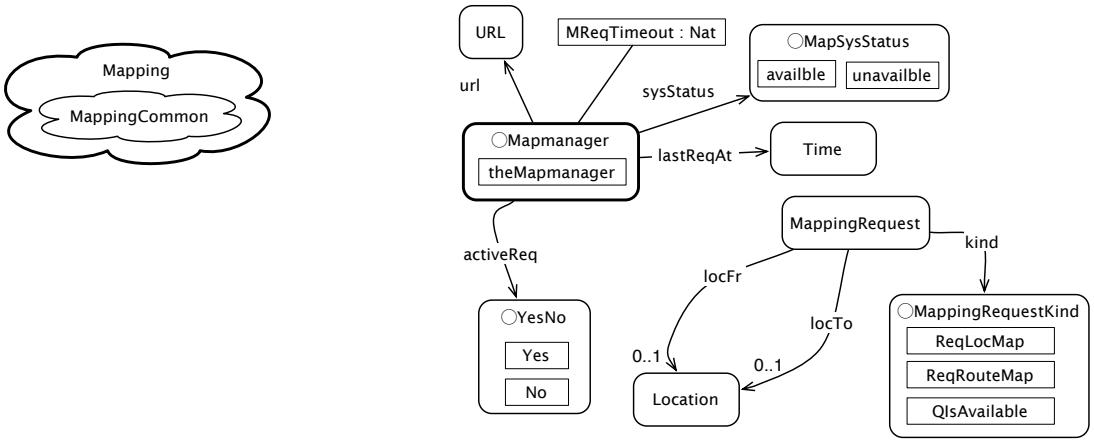


Figure 17.1: Package *Mapping* (left) and its structural diagram (right).

17.2 Behaviour

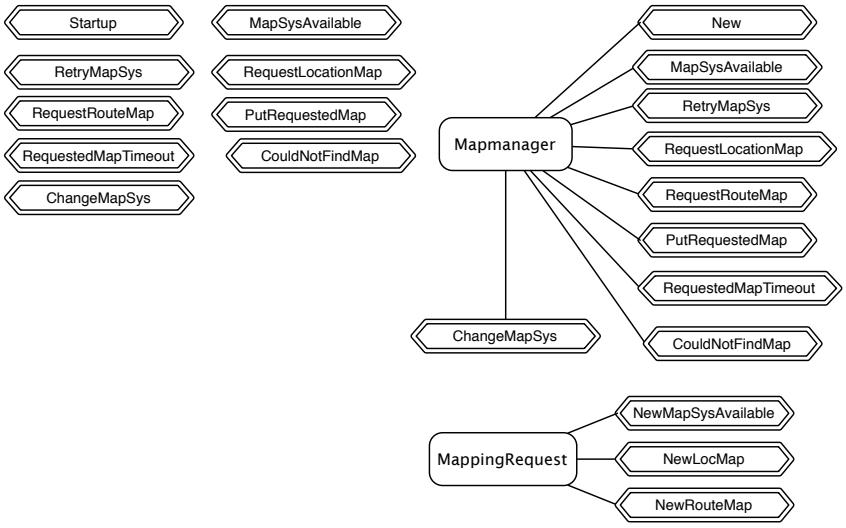


Figure 17.2: Behavioural diagram of package *Mapping*.

17.2.1 blob *MappingRequest*

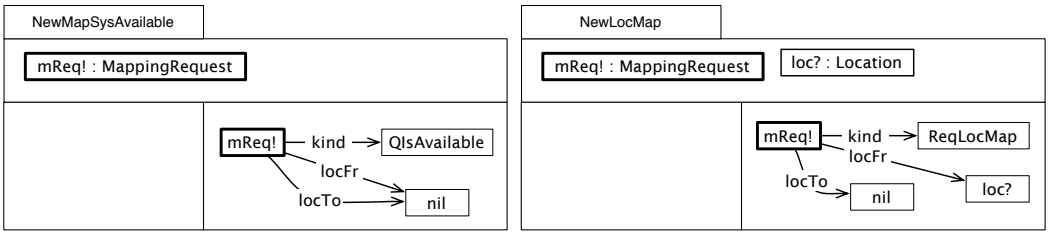


Figure 17.3: Contract diagrams of local operations *NewMapSysAvailable* and *NewLocMap* of blob *MappingRequest*.

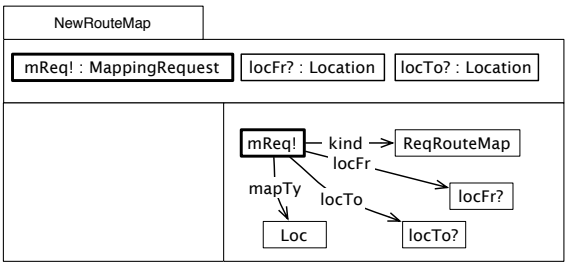


Figure 17.4: Contract diagram of local operation *NewRouteMap* blob *MappingRequest*.

17.2.2 blob *MapManager*

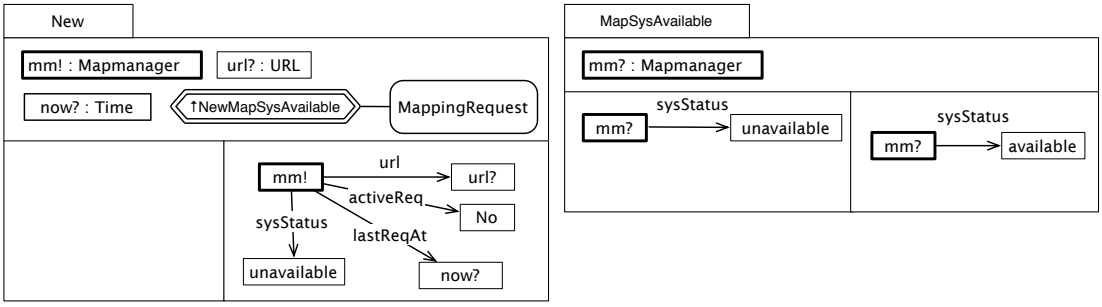


Figure 17.5: Contract diagrams for local operations *New* and *MapSysAvailable* of blob *MapManager*.

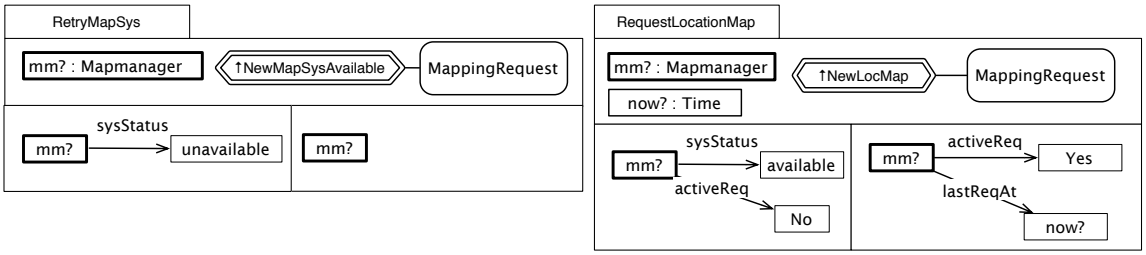


Figure 17.6: Contract diagrams for local operations *RetryMapSys* and *RequestLocationMap* of blob *MapManager*.

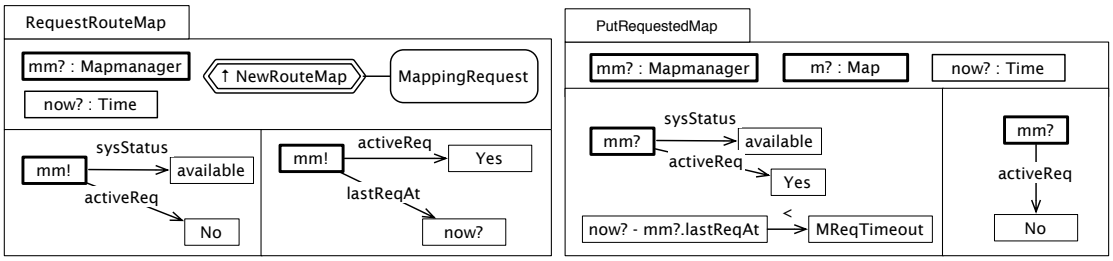


Figure 17.7: Contract diagrams for local operations *RequestRouteMap* and *PutRequestedMap* of blob *MapManager*.

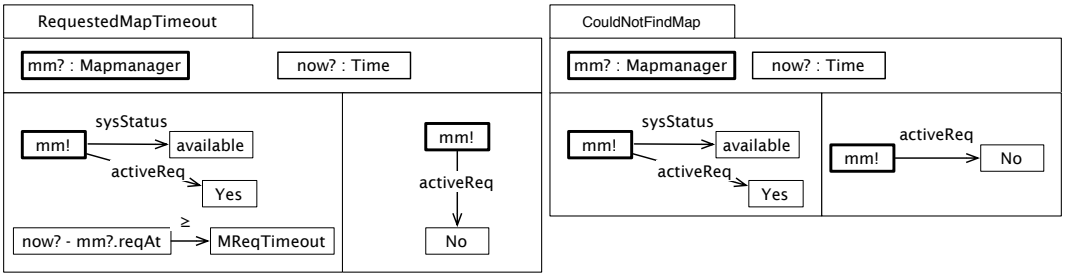


Figure 17.8: Contract diagrams for local operations *RequestedMapTimeout* and *CouldNotFindMap* of blob *MapManager*.

17.2.3 Global Behaviour

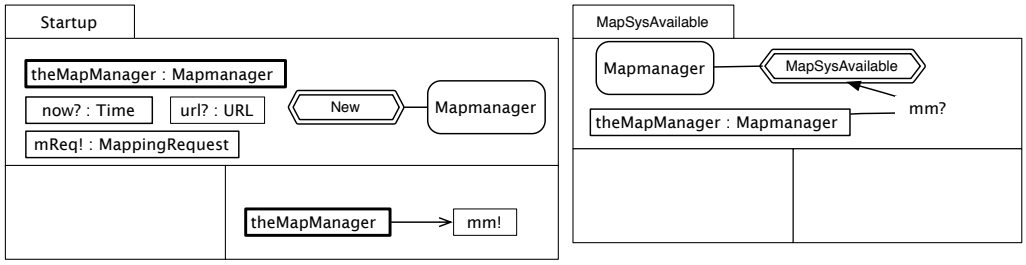


Figure 17.9: Contract diagrams of global operations *Startup* and *MapSysAvailable*.

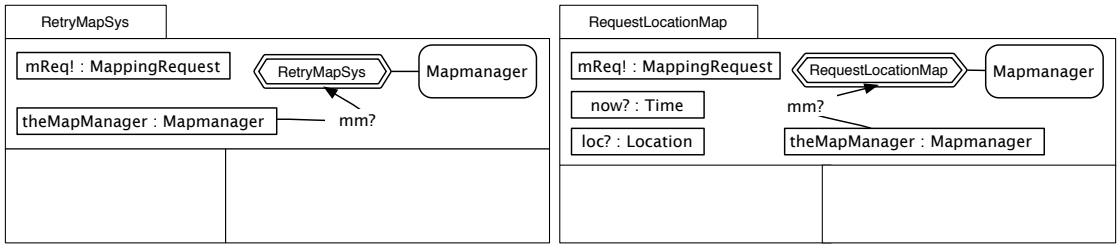


Figure 17.10: Contract diagrams of global operations *RetryMapSys* and *RequestLocationMap*.

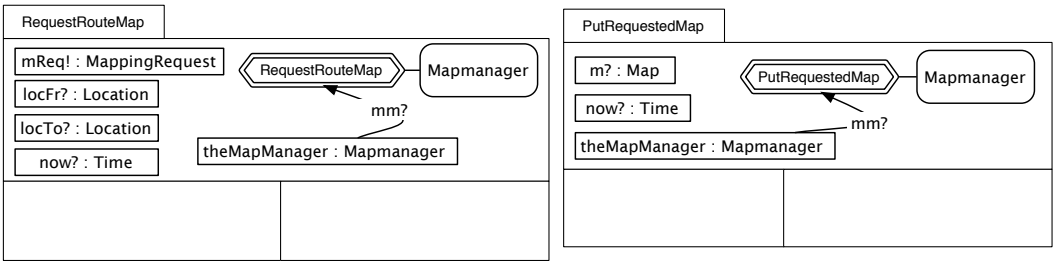


Figure 17.11: Contract diagrams of global operations *RequestRouteMap* and *PutRequestedMap*.

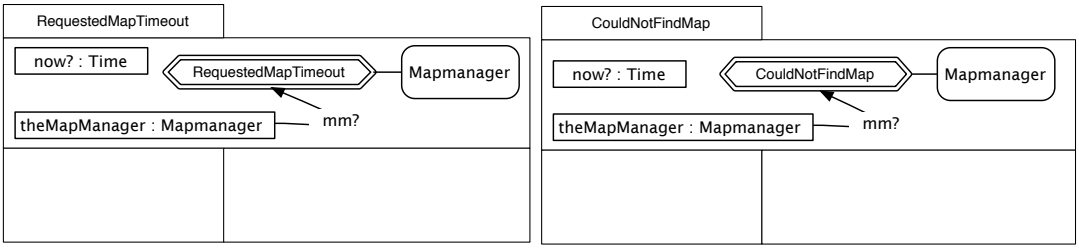


Figure 17.12: Contract diagrams of global operations *RequestedMapTimeout* and *CouldNotFindMap*.

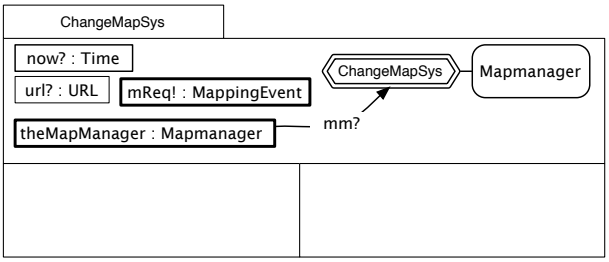


Figure 17.13: Contract diagram of global operation *ChangeMapSys*.

Chapter 18

Package *VideoSurveillanceCommon*

This chapter introduces VCL package *VideoSurveillanceCommon*, which defines structures that are common across several packages that use video surveillance functionality.

18.1 Structure

Figure 18.1: Package *VideoSurveillanceCommon* (left) and its structural diagram (right).

Chapter 19

Package *VideoSurveillance*

This chapter introduces VCL package *VideoSurveillance*, which localises the generic video surveillance concern. This is related with the handling of videos from an external video surveillance system. Package *VideoSurveillance* extends package *VideoSurveillanceCommon* (chapter 18)

19.1 Structure

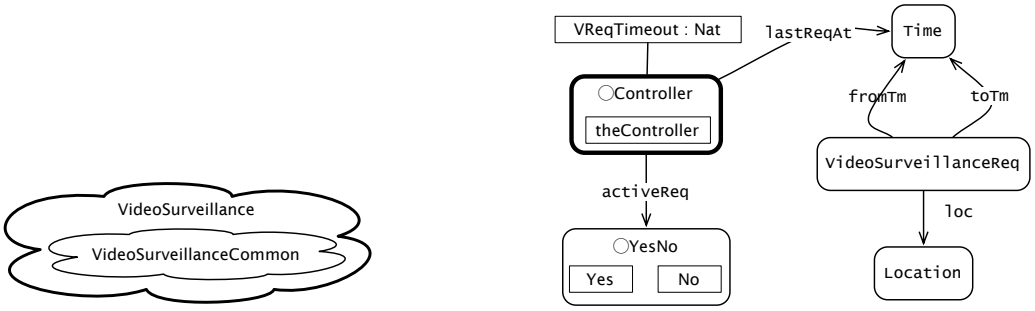


Figure 19.1: Package *VideoSurveillance* (left) and its structural diagram (right).

19.2 Behaviour

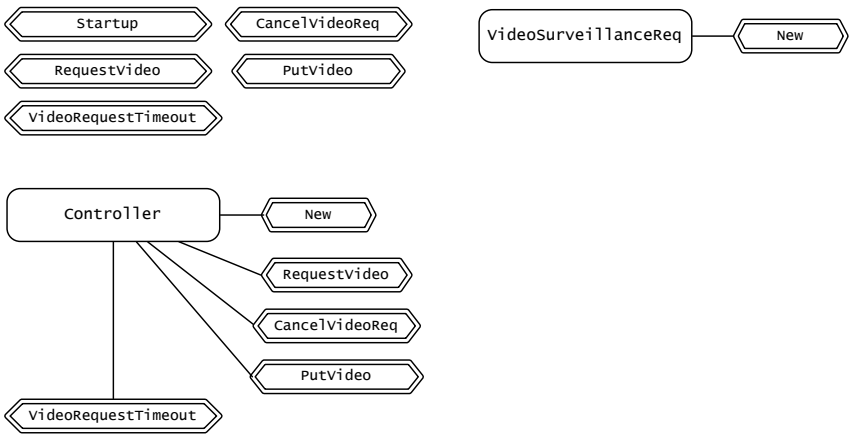


Figure 19.2: Behavioural diagram of package *VideoSurveillance*.

19.2.1 blob *VideoSurveillanceReq*

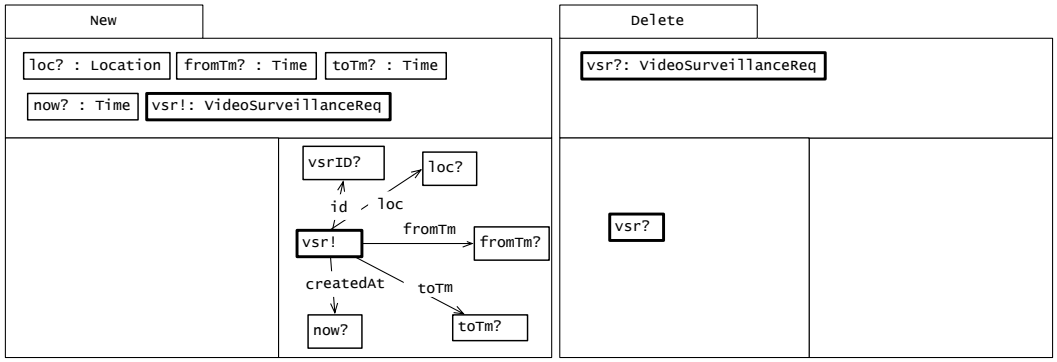


Figure 19.3: Contract diagram of local operations *New* and *Delete* of blob *VideoSurveillanceReq*.

19.2.2 blob *Controller*

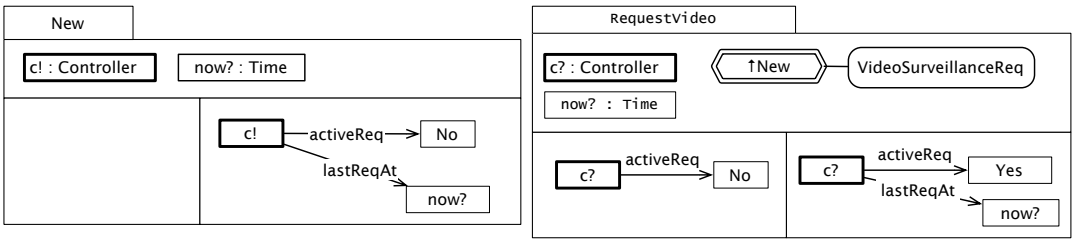


Figure 19.4: Contract diagrams for local operations *New* and *RequestVideo* of blob *Controller*.

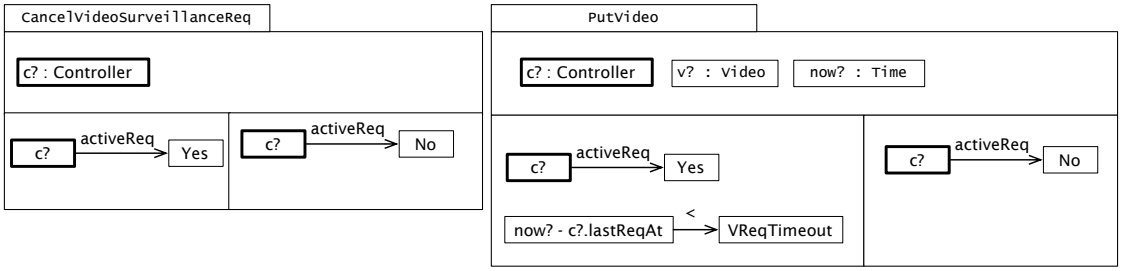


Figure 19.5: Contract diagrams for local operations *CancelVideoReq* and *PutVideo* of blob *Controller*.

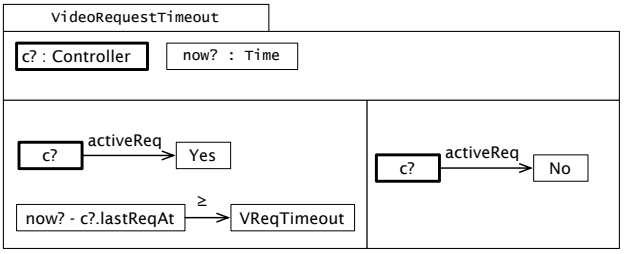


Figure 19.6: Contract diagram for local operation *VideoRequestTimeout*.

19.2.3 Global Behaviour

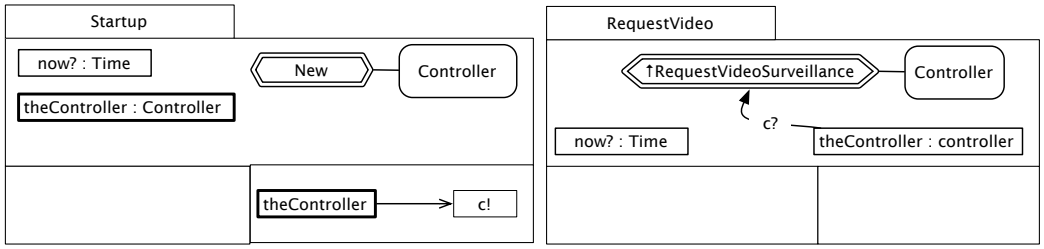


Figure 19.7: Contract diagrams of global operation *Startup* and *RequestVideo*.

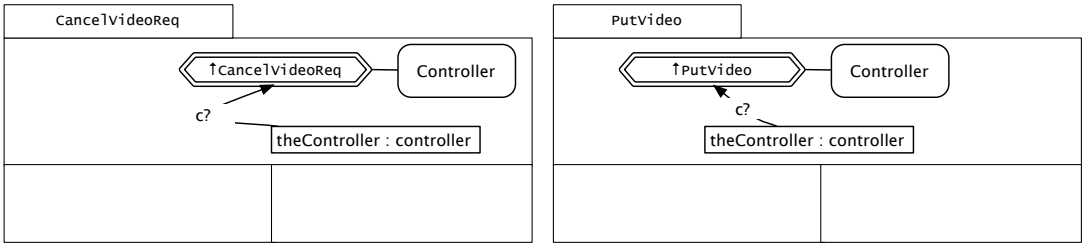


Figure 19.8: Contract diagrams of global operation *CancelVideoReq* and *PutVideo*.

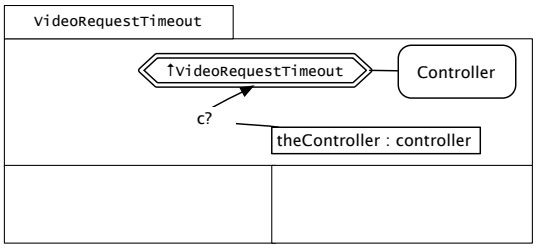


Figure 19.9: Contract diagram of global operation *VideoRequestTimeout*.

Chapter 20

Package *LocationTracking*

This chapter introduces VCL package *LocationTracking*, which localises the generic *location tracking* concern. This is related with the interaction with a GPS receiver that gives the current position of some mobile device.

20.1 Structure

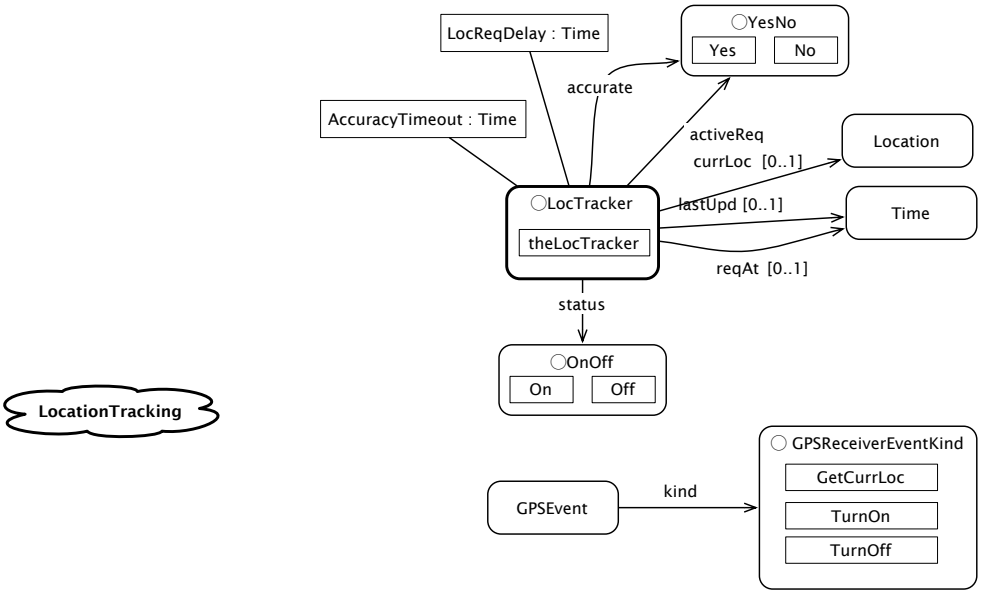


Figure 20.1: Package *LocationTracking* (left) and its structural diagram (right).

20.2 Behaviour

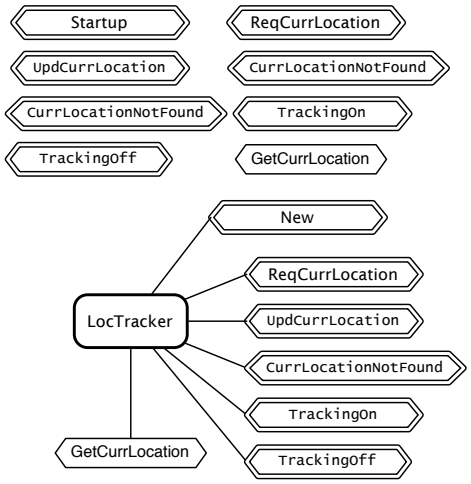


Figure 20.2: Behavioural diagram of package *LocationTracking*.

20.2.1 blob *LocTracker*

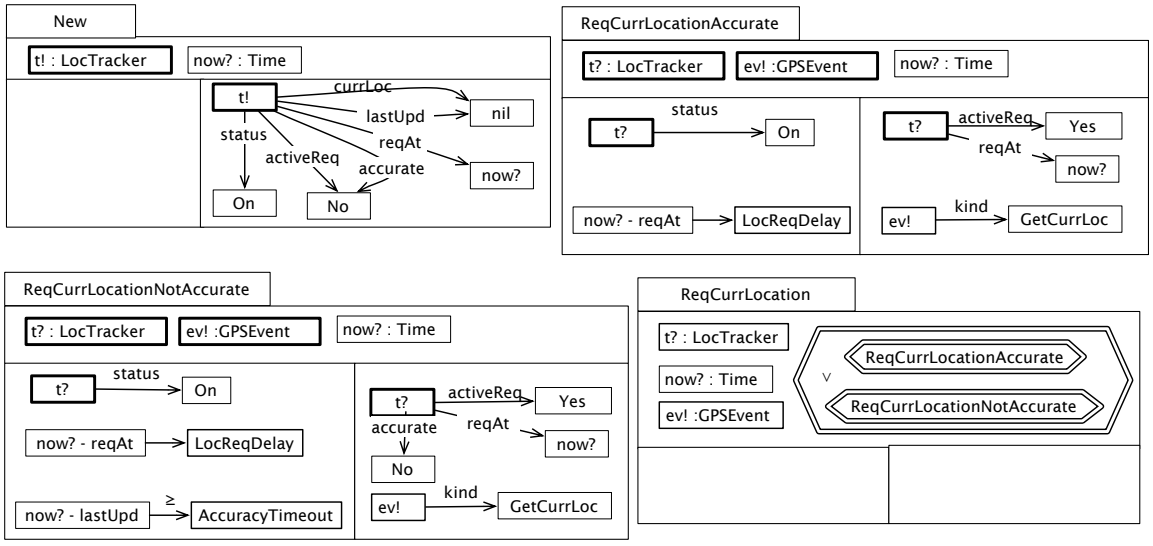


Figure 20.3: Contract diagrams for local operations *New* and *ReqCurrLocation* of blob *LocTracker*.

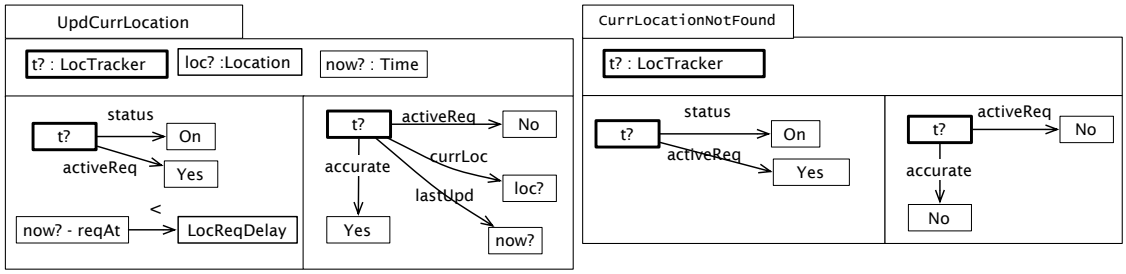


Figure 20.4: Contract diagrams for local operations *UpdCurrLocation* and *CurrLocationNotFound* of blob *LocTracker*.

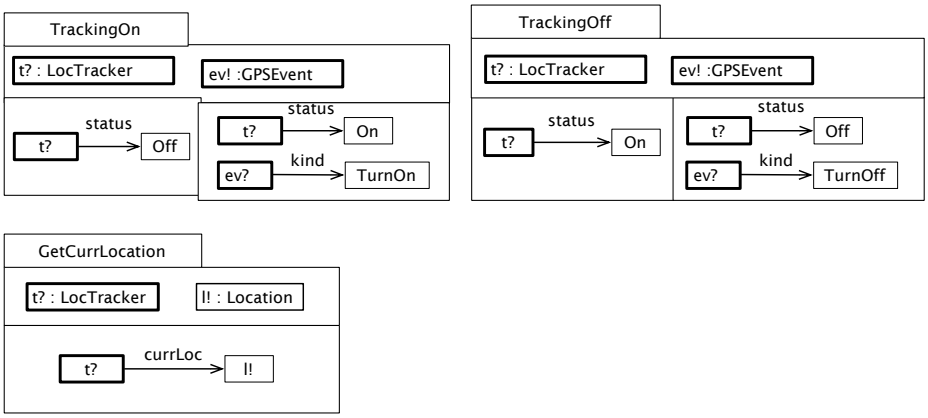


Figure 20.5: Contract and constraint diagrams for local operations *TrackingOn*, *TrackingOff* and *GetCurrLocation* of blob *LocTracker*.

20.2.2 Global Behaviour

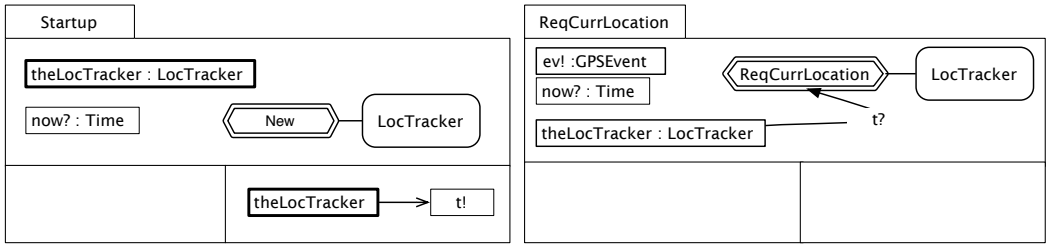


Figure 20.6: Contract diagrams of global operation *Startup* and *ReqCurrLocation*.

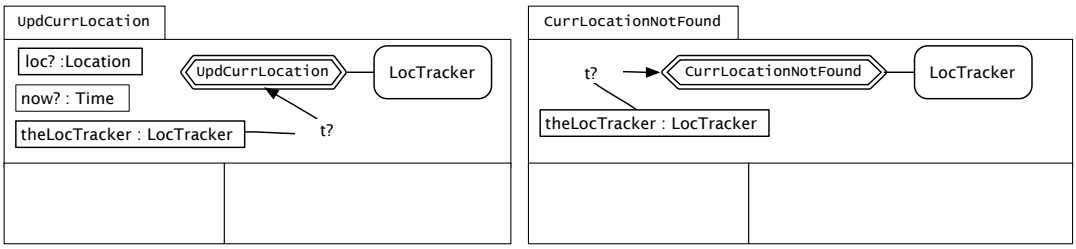


Figure 20.7: Contract diagrams of global operation *UpdCurrLocation* and *CurrLocationNotFound*.

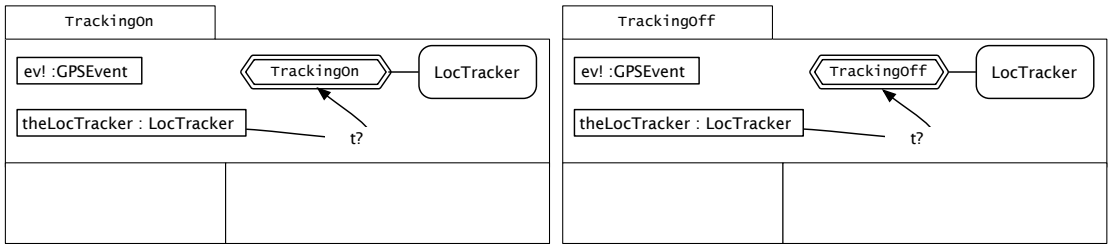


Figure 20.8: Contract diagram of global operations *TrackingOn* and *TrackingOff*.

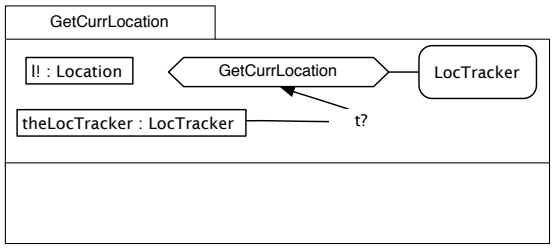


Figure 20.9: Constraint diagram of global operation *GetCurrLocation*.

Part III

Adapting Generic Packages to CCCMS context

Chapter 21

Package *ACTasksCCCMS*

This chapter introduces package *ACTasksCCCMS*, defining the tasks of CCCMS to be subject to access control. Package *ACTasksCCCMS* sees package *AccessControl* (chapter 10); it defines subsets of blob *Task* from package *AccessControl*.

21.1 Structure

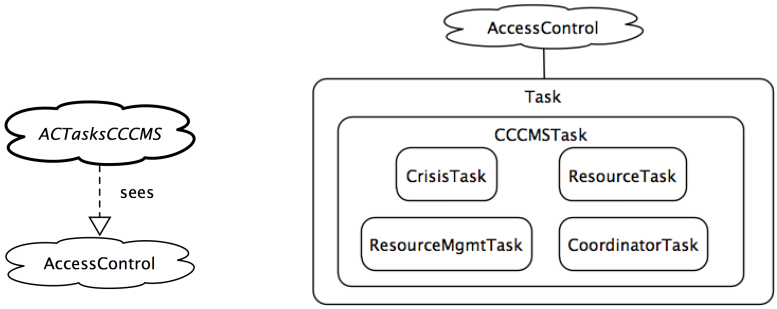


Figure 21.1: Package *ACTasksCCCMS* sees package *AccessControl* (left). Structural diagram of package *ACTasksCCCMS* (right).

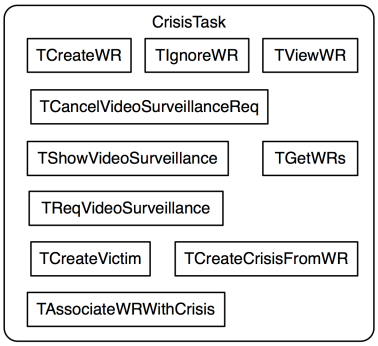


Figure 21.2: Local structural diagram of blob *CrisisTask* .

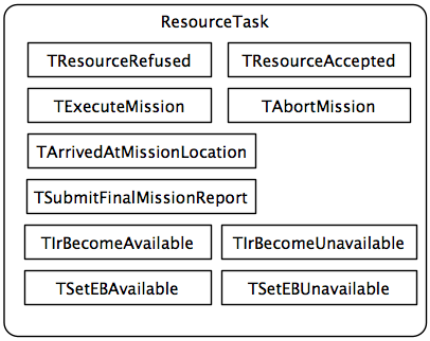


Figure 21.3: Local structural diagram of blob *ResourceTask* .

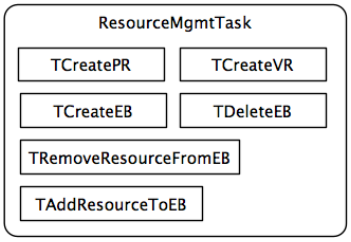


Figure 21.4: Local structural diagram of blob *ResourceMgmtTask* .

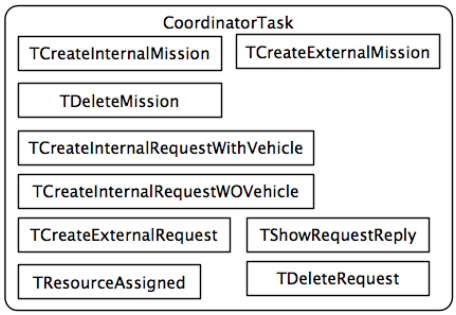


Figure 21.5: Local structural diagram of blob *CoordinatorTask* .

Chapter 22

Package *AuthorisationCCCMS*

This chapter introduces package *AuthorisationCCCMS*, which does the customisation of package *Authorisation* to the CCCMS context. Package *AuthorisationCCCMS* sees package *ACTasksCCCMS* (chapter 21) and extends package *Authorisation* (chapter 12).

22.1 Structure

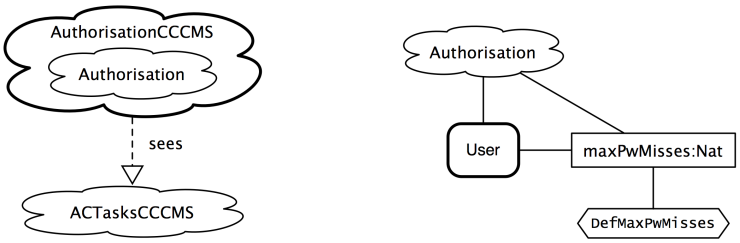


Figure 22.1: Package *AuthorisationCCCMS* extends *Authorisation* and sees *ACTasksCCCMS* (left). Structural diagram of package *AuthorisationCCCMS* (right) .

22.2 Configuration by defining an initial state

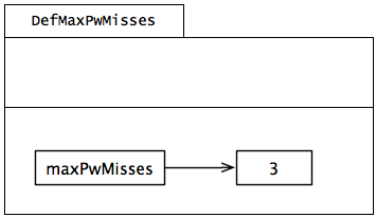


Figure 22.2: Constraint diagram defining constant *maxPwMisses* to value 3.

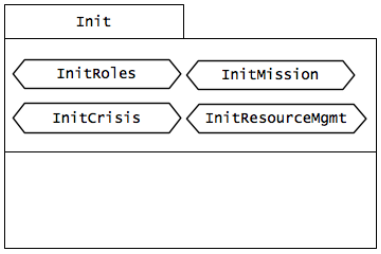


Figure 22.3: Constraint diagram defining initial state of package.

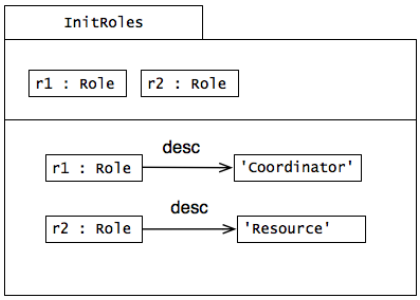


Figure 22.4: Constraint diagram defining constraint *InitialRoles* part of initial state.

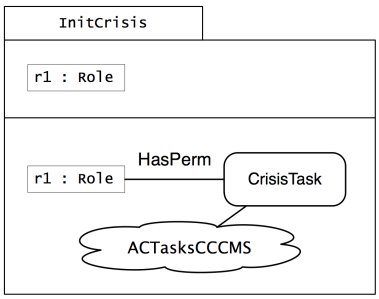


Figure 22.5: Constraint diagram defining constraint *InitCrisis* part of initial state.

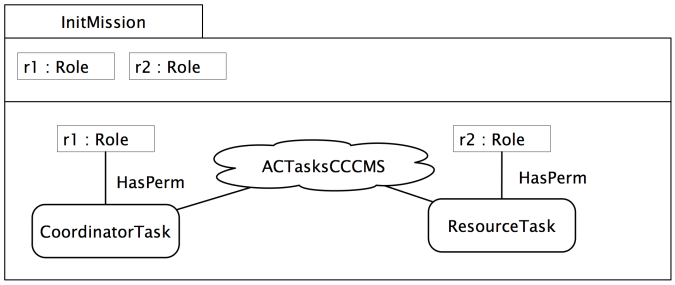


Figure 22.6: Constraint diagram defining constraint *InitMission* part of initial state.

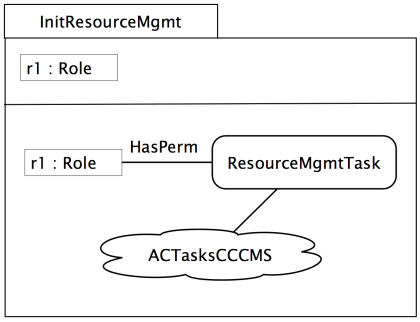


Figure 22.7: Constraint diagram defining constraint *InitResources* part of initial state.

22.3 Behaviour

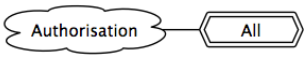


Figure 22.8: Behavioural diagram of package *AuthorisationCCCMS*.

Chapter 23

Package *EventsCCCMS*

This chapter introduces package *EventsCCCMS*, which defines events to be logged in CCCMS. Package *EventsCCCMS* sees package *Logging* (chapter 15); it defines subsets of blob *EventKind* of *Logging*, defining events to be logged.

23.1 Structure

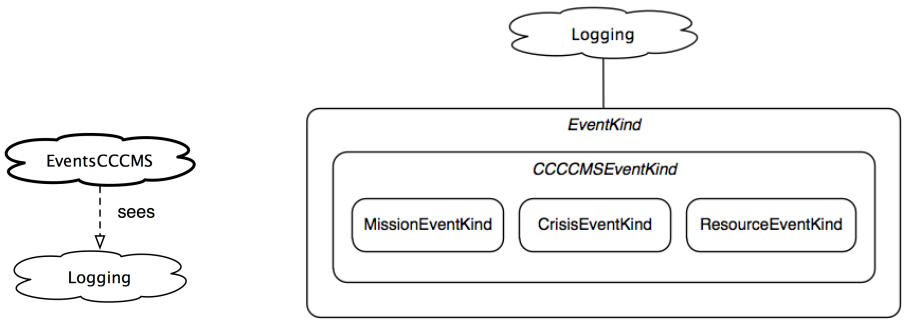


Figure 23.1: Package *EventsCCCMS* sees package *Logging* (left). Structural diagram of package *EventsCCCMS* (right).

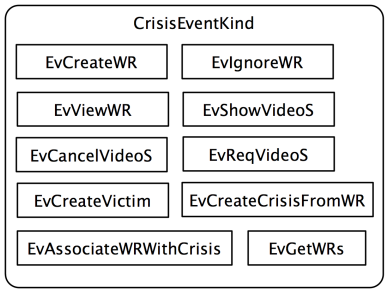


Figure 23.2: Local structural diagram of blob *CrisisEventKind* .

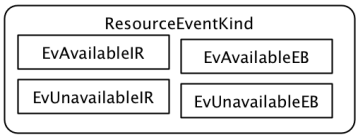


Figure 23.3: Local structural diagram of blob *ResourceEventKind* .

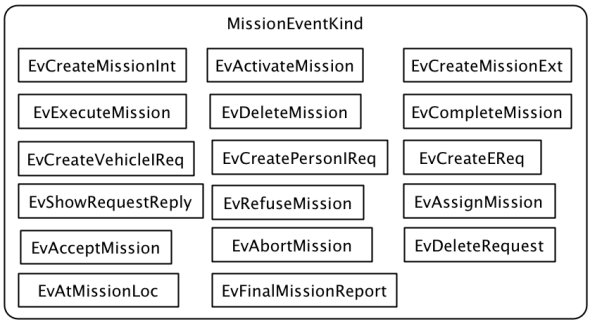


Figure 23.4: Local structural diagram of blob *MissionEventKind* .

Chapter 24

Package *LoggingCCCMS*

This chapter introduces package *LoggingCCCMS*, which customises package *Logging* to the CCCMS context. Package *LoggingCCCMS* extends package *Logging* (see chapter 15) and see package *EventsCCCMS* (chapter 23).

24.1 Structure

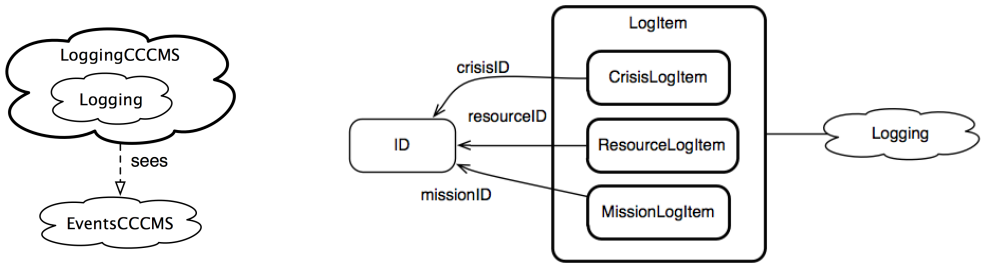


Figure 24.1: Package *LoggingCCCMS* extends package *Logging* and sees package *EventsCCCMS* (left). Structural diagram of package *LoggingCCCMS* (right).

24.2 Behaviour

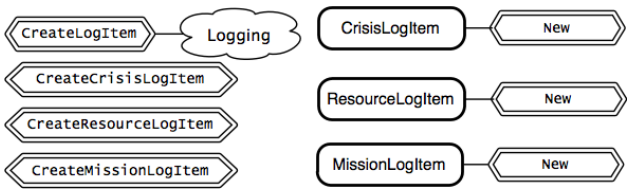


Figure 24.2: Behavioural diagram of package *LoggingCCCMS*.

24.2.1 Blob *CrisisLogItem*

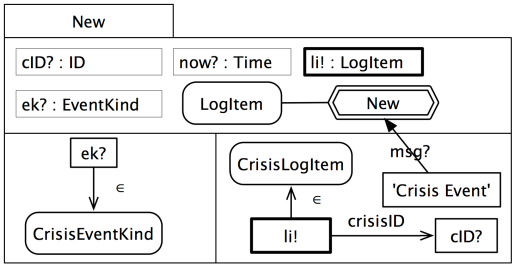


Figure 24.3: Contract diagram of local operation *CrisisLogItem.New*.

24.2.2 Blob *ResourceLogItem*

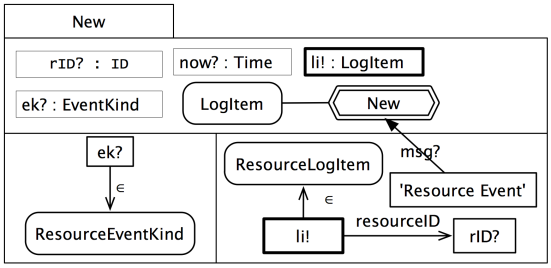


Figure 24.4: Contract diagram of local operation *ResourceLogItem.New*.

24.2.3 Blob *MissionLogItem*

24.2.4 Global Behaviour

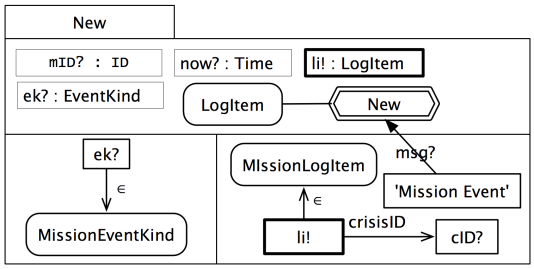


Figure 24.5: Contract diagram of local operation *MissionLogItem.New*.

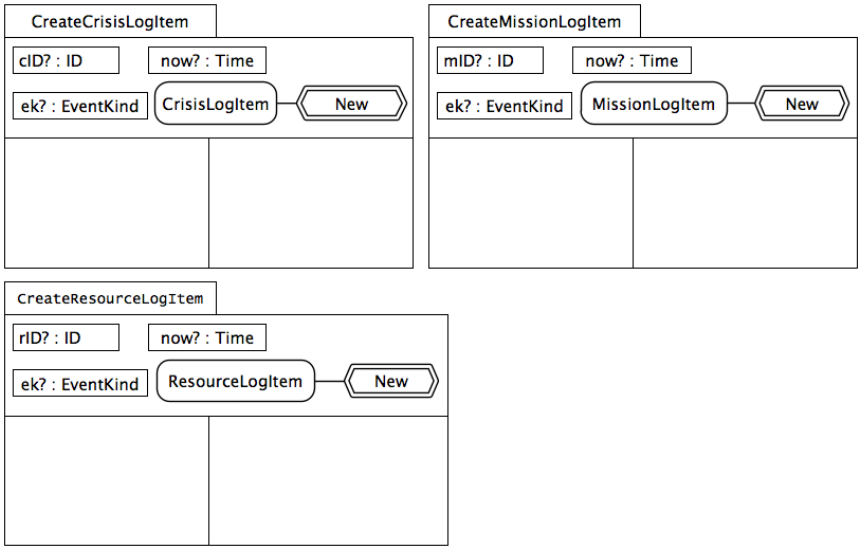


Figure 24.6: Contract diagrams for global operations *CreateCrisisLogItem*, *CreateMissionLogItem* and *CreateResourceLogItem*.

Chapter 25

Package *SessionMgmtCCCMS*

This chapter introduces package *SessionMgmtCCCMS*, which customises package *SessionMgmt* to the CCCMS context. Package *SessionMgmtCCCMS* extends package *SessionMgmt* (chapter 9). This package is extended in packages *CrisisWithAspects* (chapter 43) and *MissionWithAspects* (chapter 44).

25.1 Structure

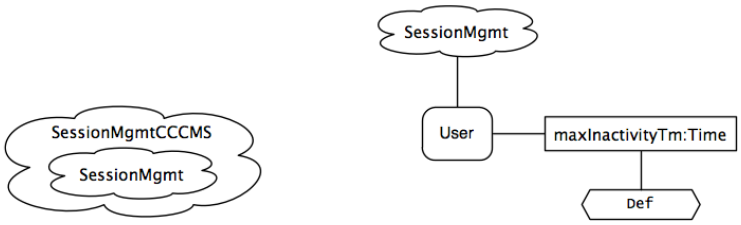


Figure 25.1: Package *SessionMgmtCCCMS* extends package *SessionMgmt* (left). Structural diagram of package *SessionMgmtCCCMS* (right).

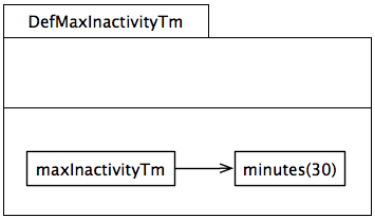


Figure 25.2: Constraint diagram defining constant *maxInactivityTm* of blob *Session* to *minutes*(30).

25.2 Behaviour

Figure 25.3: Behavioural diagram of package *SessionMgmtCCCMS*.

Chapter 26

Package *MappingCCCMS*

This chapter introduces package *MappingCCCMS*, which customises package *Mapping* to the CCCMS context. Package *MappingCCCMS* extends package *Mapping* (chapter 17).

26.1 Structure

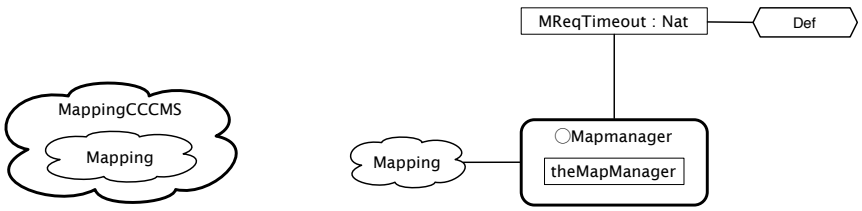


Figure 26.1: Package *MappingCCCMS* extends package *Mapping* (left). Structural diagram of package *MappingCCCMS* (right).

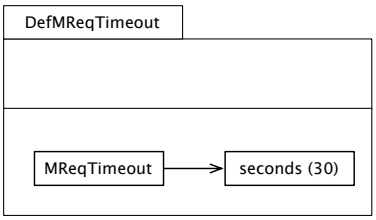


Figure 26.2: Constraint diagram defining constant *MReqTimeout* of blob *MapManager* to 30 seconds.

26.2 Behaviour

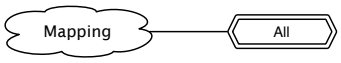


Figure 26.3: Behavioural diagram of package *MappingCCCMS*, which defines all its operations by integral extension of all operations of package *Mapping*.

Chapter 27

Package *VideoSurveillanceCCCMS*

This chapter introduces package *VideoSurveillanceCCCMS*, which customises package *VideoSurveillance* to the CCCMS context. Package *VideoSurveillanceCCCMS* extends package *VideoSurveillance* (chapter 19).

27.1 Structure

Figure 27.1: Package *VideoSurveillanceCCCMS* extends package *VideoSurveillance* (left). Structural diagram of package *VideoSurveillanceCCCMS* (right).

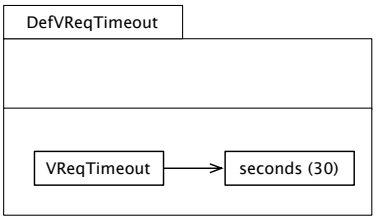


Figure 27.2: Constraint diagram defining constant *VReqTimeout* of blob *Controller* to 30 seconds.

27.2 Behaviour

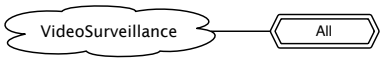


Figure 27.3: Behavioural diagram of package *VideoSurveillanceCCCMS*, which defines all its operations by integral extension of all operations of package *VideoSurveillance*.

Part IV

Generic Domain Packages

Chapter 28

Package *MappingDisplay*

This chapter introduces package *MappingDisplay*, which addresses the mapping display concerns of both *CentralCCCMS* and *MobCCCMS* subsystems. Package *MappingDisplay* extends packages *MappingCommon* (chapter 16).

28.1 Structure

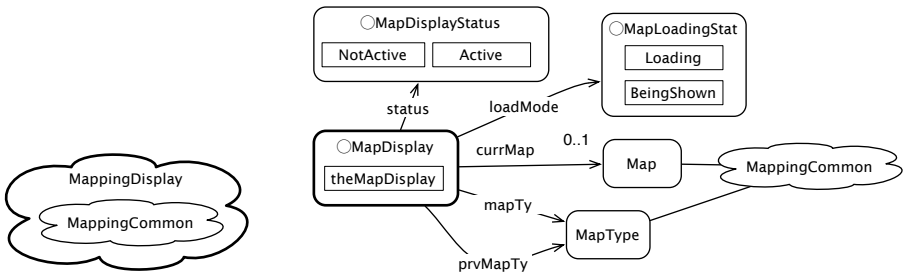


Figure 28.1: Package (left) and structural (right) diagram of package *MappingDisplay*.

28.2 Behaviour

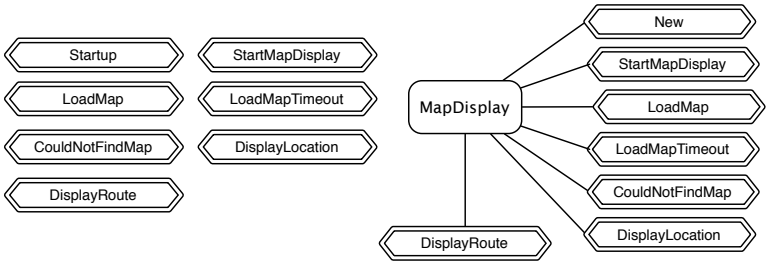


Figure 28.2: Behaviour diagram of package *MappingDisplay*.

28.2.1 Blob *MapDisplay*

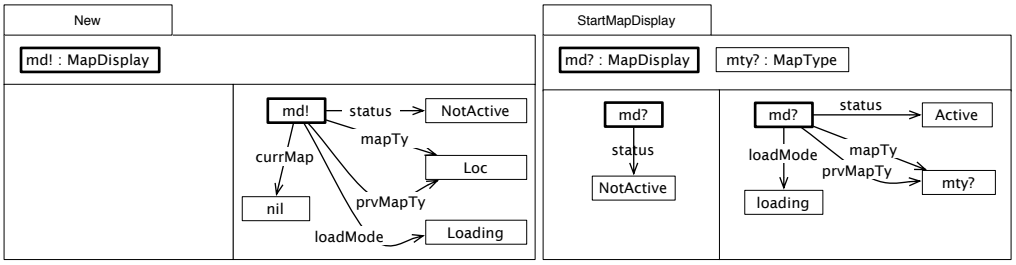


Figure 28.3: Contract diagrams for local operations *New* and *StartMapDisplay* of blob *MapDisplay*.

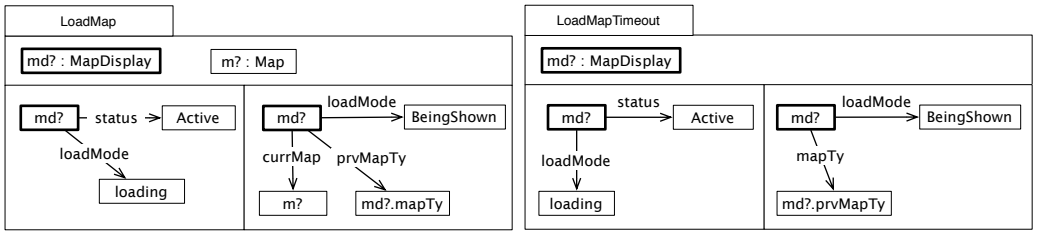


Figure 28.4: Contract diagrams for local operations *LoadMap* and *LoadMapTimeout* of blob *MapDisplay*.

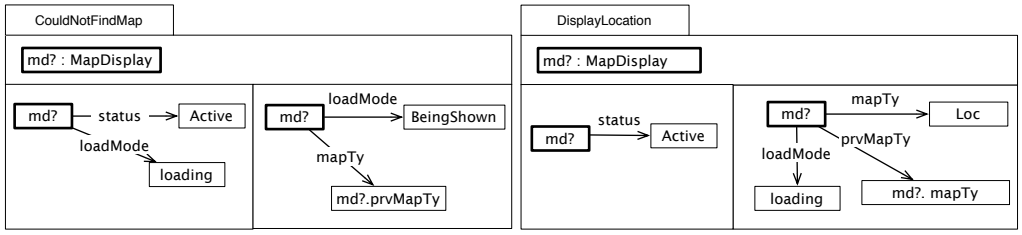


Figure 28.5: Contract diagrams for local operations *CouldNotFindMap* and *DisplayLocation* of blob *MapDisplay*.

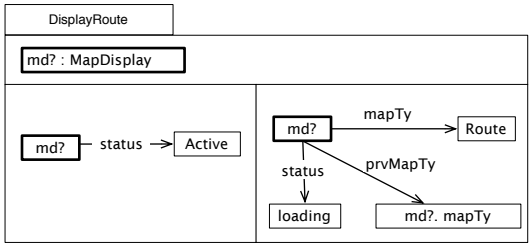


Figure 28.6: Contract diagram for local operations *DisplayRoute* of blob *MapDisplay*.

28.2.2 Global Behaviour

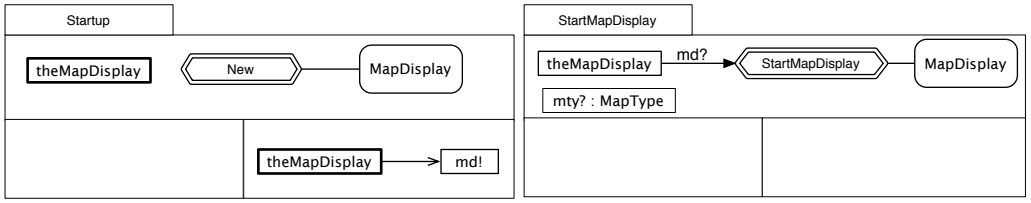


Figure 28.7: Contract diagrams for global operations *Startup* and *StartMapDisplay*.

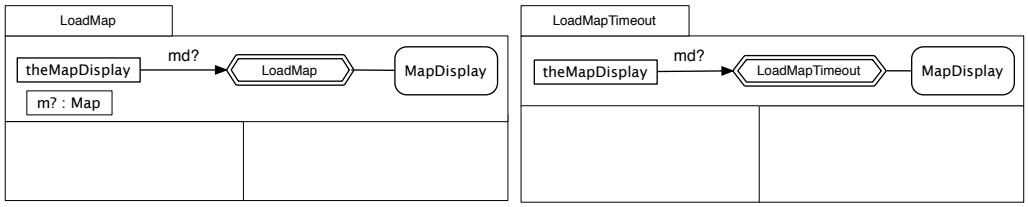


Figure 28.8: Contract diagrams for global operations *LoadMap* and *LoadMapTimeout*.

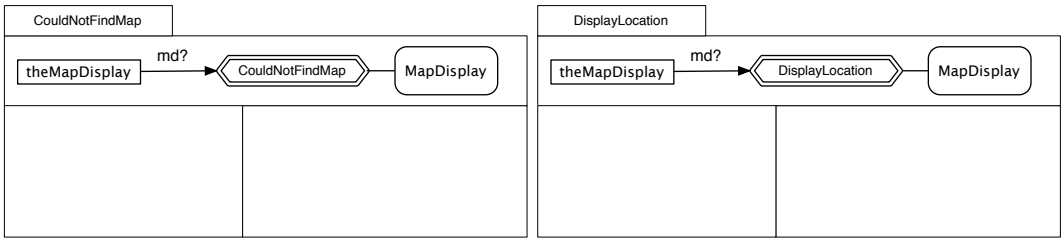


Figure 28.9: Contract diagrams for global operations *CouldNotFindMap* and *DisplayLocation*.

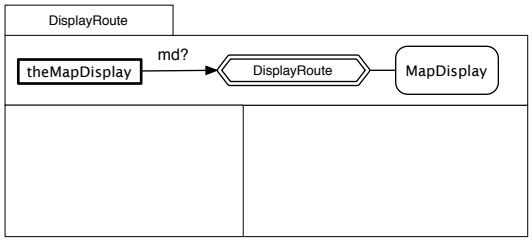


Figure 28.10: Contract diagram for global operation *DisplayRoute*.

Chapter 29

Package *MappingDisplayWithMapSys*

This chapter introduces package *MappingDisplayWithMapSys*, which adds the mapping generic concern (that deals with the mapping system) to the the package *MappingDisplay*. Package *MappingDisplayWithMapSys* extends packages *MappingDisplay* (chapter 28) and *MappingCCCMS* (chapter 26).

29.1 Structure

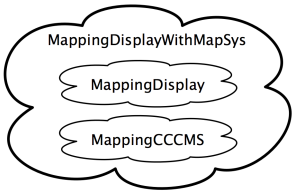


Figure 29.1: Package diagram of package *MappingDisplayWithMapSys*.

29.2 Behaviour

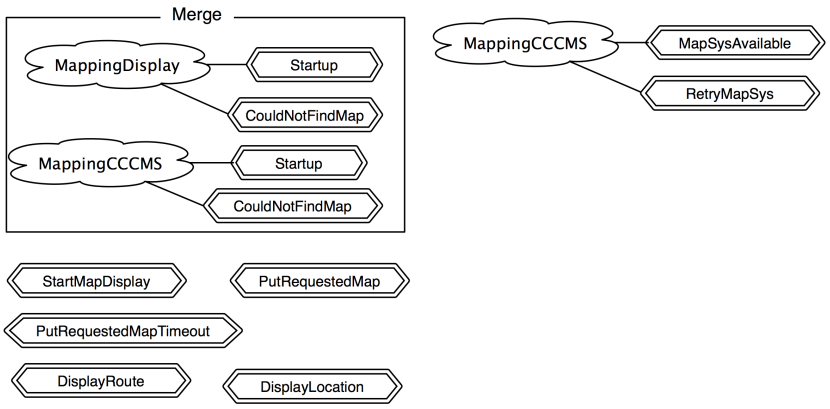


Figure 29.2: Behaviour diagram of package *MappingDisplayWithMapSys*.

29.2.1 Global Behaviour

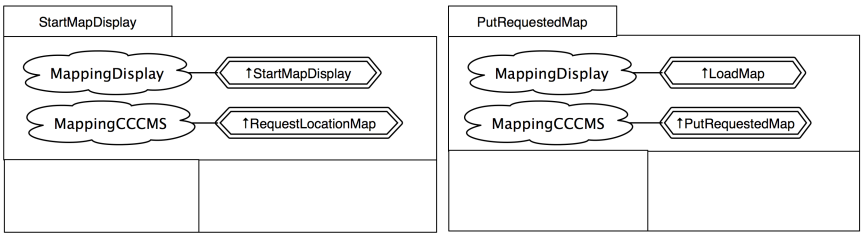


Figure 29.3: Contract diagrams for global operations *StartMapDisplay* and *PutRequestedMap*.

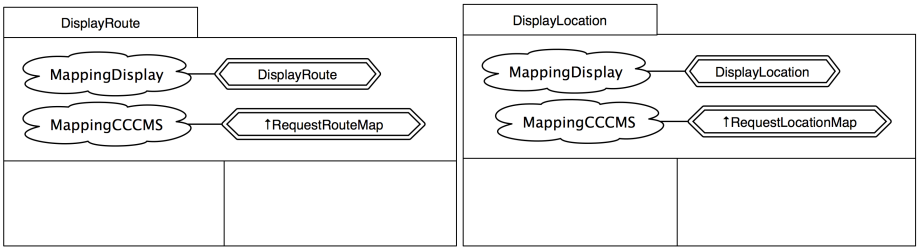


Figure 29.4: Contract diagrams for global operations *DisplayRoute* and *DisplayLocation*.

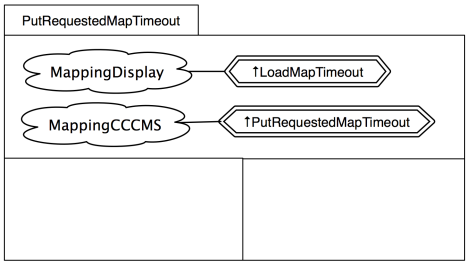


Figure 29.5: Contract diagrams for global operation *PutRequestedMapTimeout*.

Part V

Domain Packages

Chapter 30

Package *WitnessReports*

This chapter introduces domain package *WitnessReport*, which encapsulates functionality related with witness reports in the CCCMS.

30.1 Structure

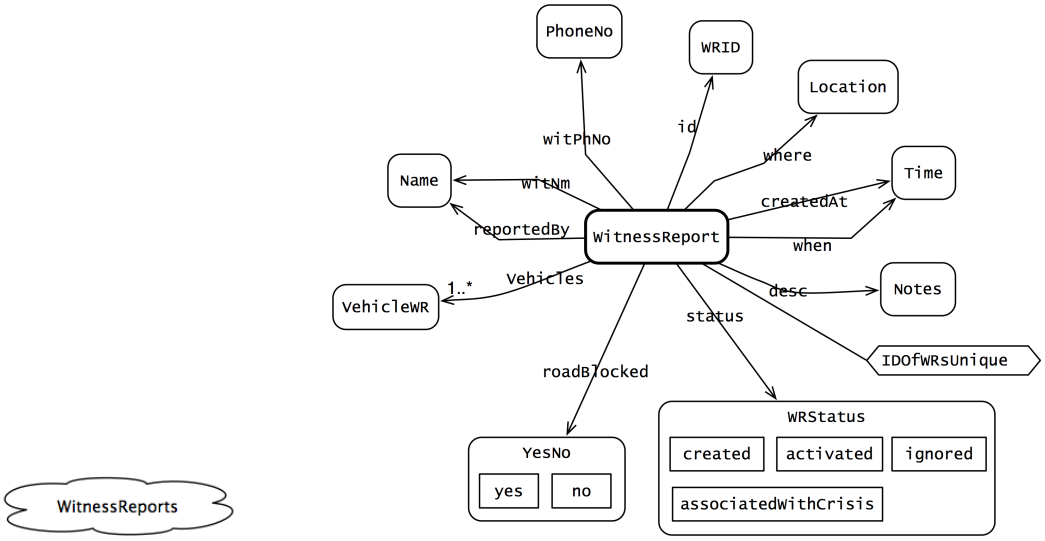


Figure 30.1: Package *WitnessReports* (left). Global view of structural diagram of package *WitnessReports* (right).

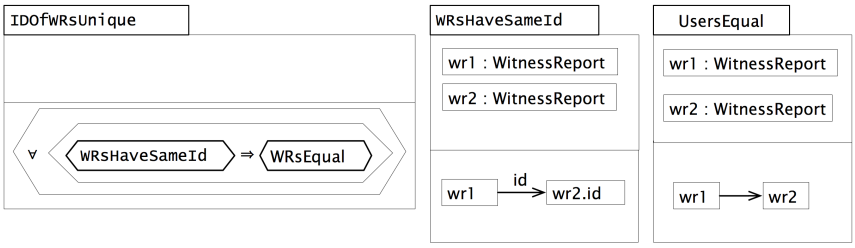


Figure 30.2: Constraint diagram describing *WitnessReport* blob's local invariant *IDOfWRsUnique*.

30.2 Behaviour

Figure 30.3: Global view of behavioural diagram of package *WitnessReports*.

30.2.1 Blob *WitnessReport*

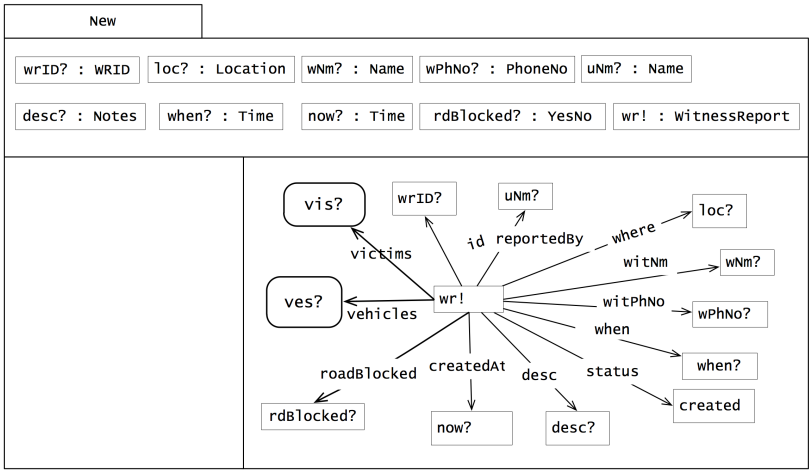


Figure 30.4: Contract diagram of local operation *New* of blob *WitnessReport*.

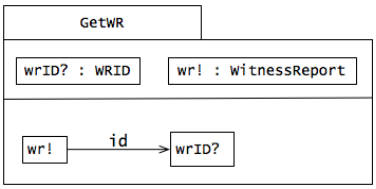


Figure 30.5: Constraint diagram describing behavioural constraint *GetWR* of blob *WitnessReport*.

30.2.2 Global Behaviour

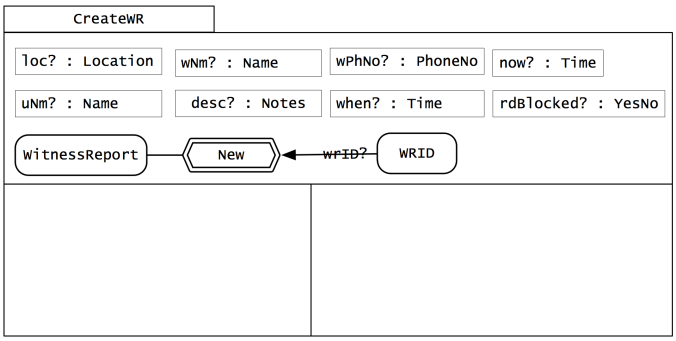


Figure 30.6: Contract diagram describing global operation *CreateWR*.

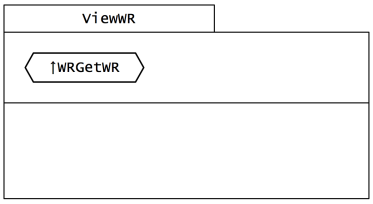


Figure 30.7: Contract diagram describing global operation *ViewWR*.

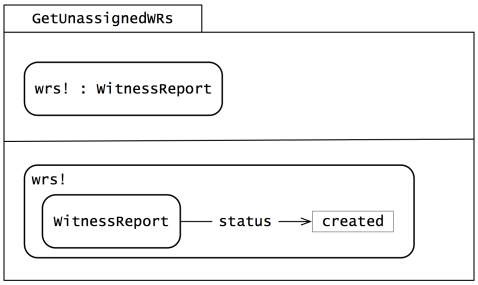


Figure 30.8: Contract diagram describing global operation *GetUnassignedWRs*.

Chapter 31

Package *CrisisCommon*

This chapter introduces package *CrisisCommon*, which encapsulates sets that are related with crisis.

31.1 Structure

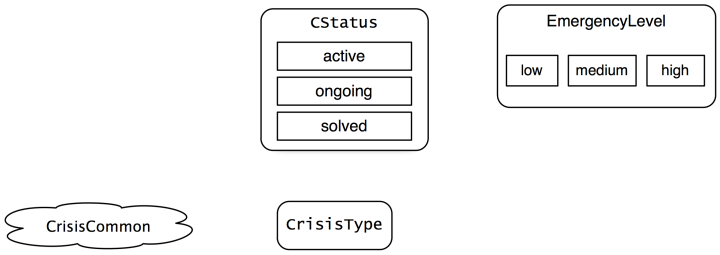


Figure 31.1: Package *CrisisCommon* (left). Structural diagram of package *CrisisCommon* (right).

Chapter 32

Package *Crisis*

This chapter introduces package *Crisis*, which encapsulates the concerns associated with the *crisis management* architectural unit (see Fig. 2.3). Package *Crisis* extends packages *WitnessReports* (chapter 30), *CrisisCommon* (chapter 31) and *VideoSurveillanceCommon* (chapter 18).

32.1 Structure

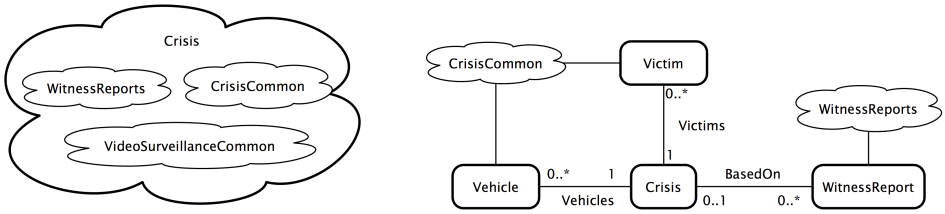


Figure 32.1: Package *Crisis* (left). Global view of structural diagram of package *Crisis* (right).

32.1.1 Blob *Crisis*

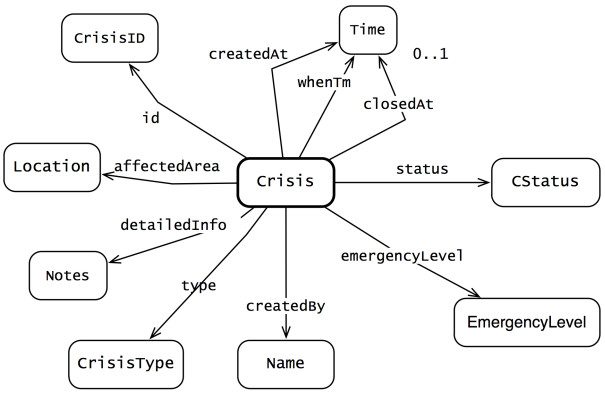


Figure 32.2: Local structural diagram of *Crisis* blob.

32.2 Behaviour

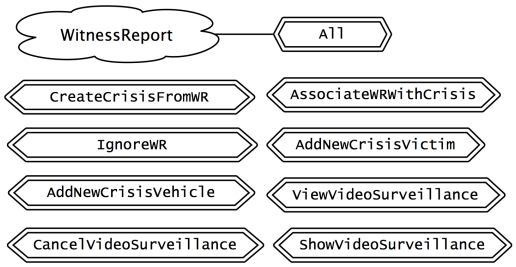


Figure 32.3: Global behaviour diagram of package *Crisis*. All global operations of package *WitnessReport* are imported as global operations of package *Crisis* using integral extension.

32.2.1 Blob *WitnessReport*

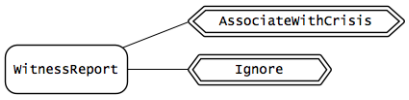


Figure 32.4: Local behavioural diagram of blob *WitnessReport* in package *Crisis*.

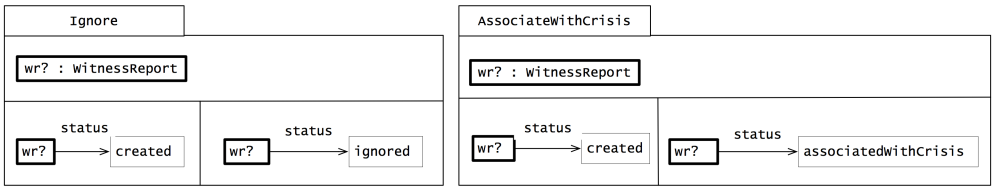


Figure 32.5: Contract diagrams for local operations *Ignore* and *AssociateWithCrisis* of blob *WitnessReport*.

32.2.2 Blob *Crisis*

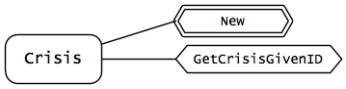


Figure 32.6: Local behavioural diagram of *Crisis* Blob.

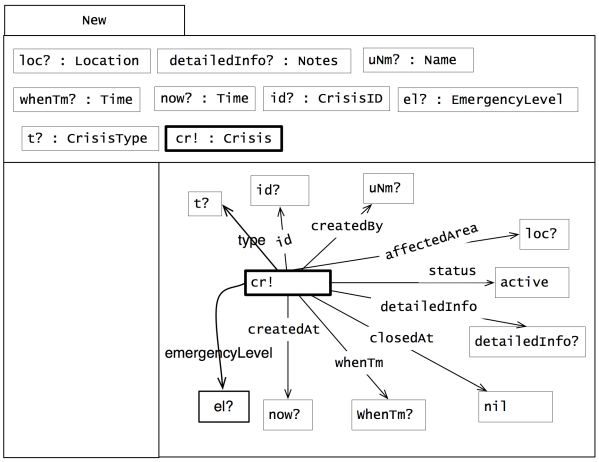


Figure 32.7: Contract diagram of local operation *New* of blob *Crisis*.

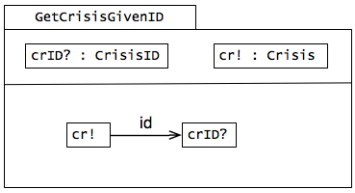


Figure 32.8: Contract diagram of local operation *GetCrisisGivenID* of blob *Crisis*.

32.2.3 Relational-edge *BasedOn*

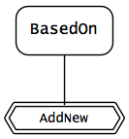


Figure 32.9: Local behavioural diagram of *BasedOn* Relational edge.

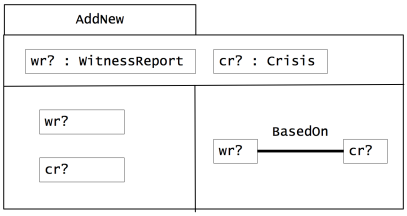


Figure 32.10: Contract diagram of operation *New* of *BasedOn* relational edge.

32.2.4 Relational-edge *Victims*

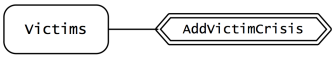


Figure 32.11: Local behavioural diagram of *Victims* Relational edge.

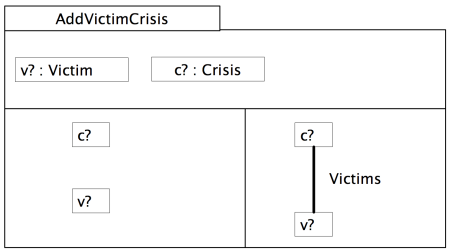


Figure 32.12: Contract diagram of local operation *AddVictimCrisis* of relational edge *Victims*.

32.2.5 Relational-edge *Vehicles*

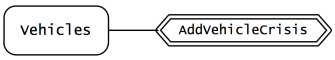


Figure 32.13: Local behavioural diagram of *Vehicles* Relational edge.

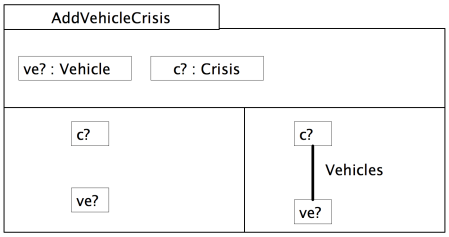


Figure 32.14: Contract diagram of local operation *AddVehicleCrisis* of relational edge *Vehicle*.

32.2.6 Global Behaviour

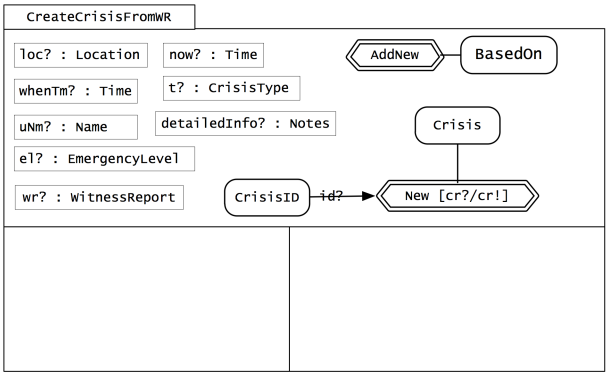


Figure 32.15: Contract diagram of global operation *CreateCrisisFromWR*.

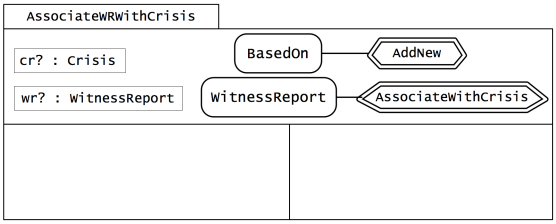


Figure 32.16: Contract diagram of global operation *AssociateWRWithCrisis*.

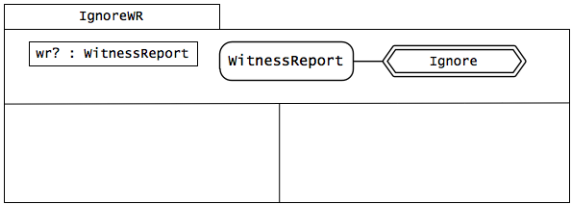


Figure 32.17: Contract diagram of global operation *IgnoreWR*.

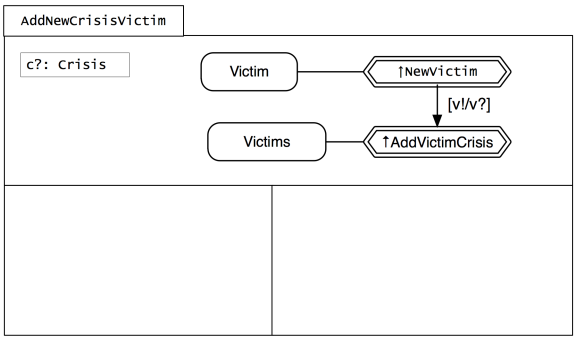


Figure 32.18: Contract diagram of global operation *AddNewCrisisVictim*.

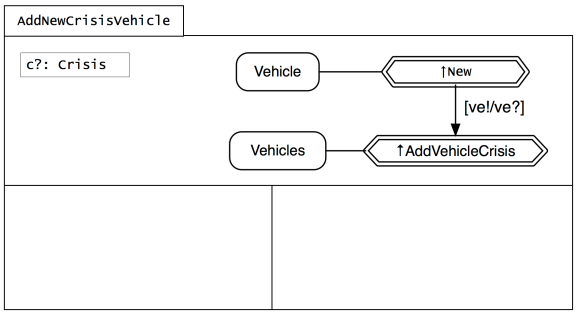


Figure 32.19: Contract diagram of global operation *AddNewCrisisVehicle*.

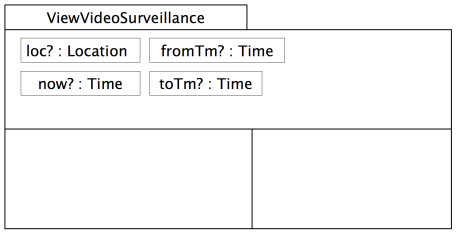


Figure 32.20: Contract diagram of global operation *ViewVideoSurveillance*.

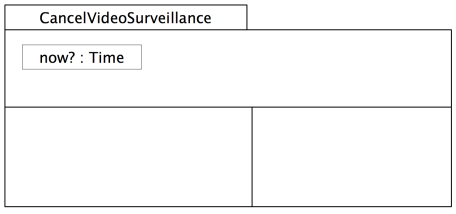


Figure 32.21: Contract diagram of global operation *CancelVideoSurveillance*.

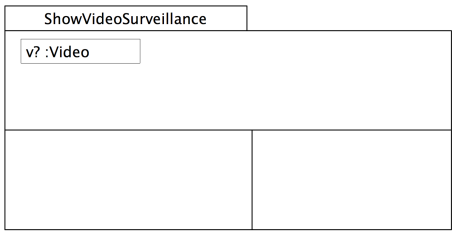


Figure 32.22: Contract diagram of global operation *ShowVideoSurveillance*.

Chapter 33

Package *Resources*

This chapter introduces package *Resources*, which address a concern related with management of resources in CCCMS.

33.1 Structure

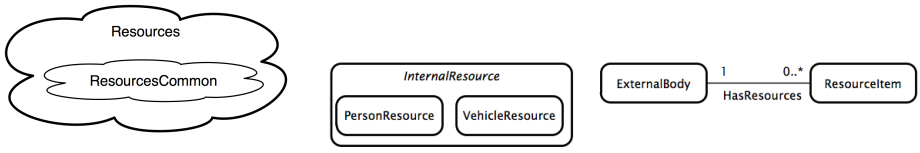


Figure 33.1: Package *Resources* (left). Global view of structural diagram of package *Resources* (right).

33.1.1 InternalResource

33.1.2 PersonResource

33.1.3 VehicleResource

33.1.4 ExternalBody

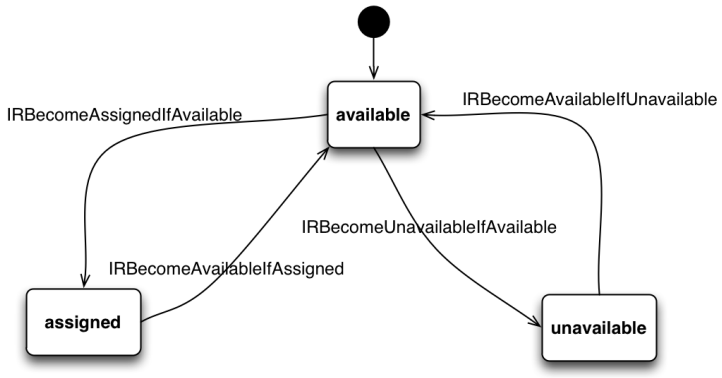


Figure 33.2: State diagram of *InternalResource* object.

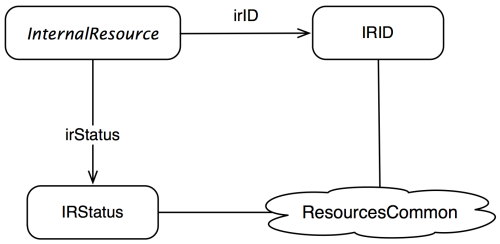


Figure 33.3: Local structural diagram of *InternalResource* blob.

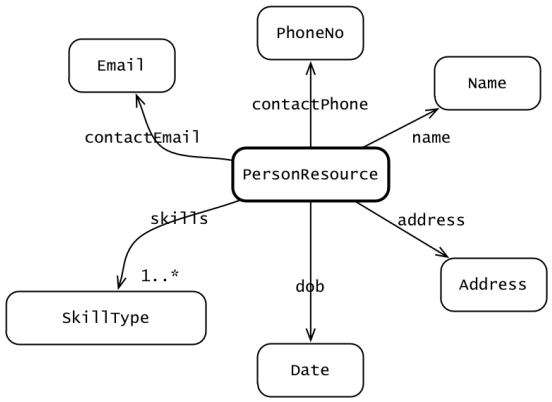


Figure 33.4: Local structural diagram of *PersonResource* blob.

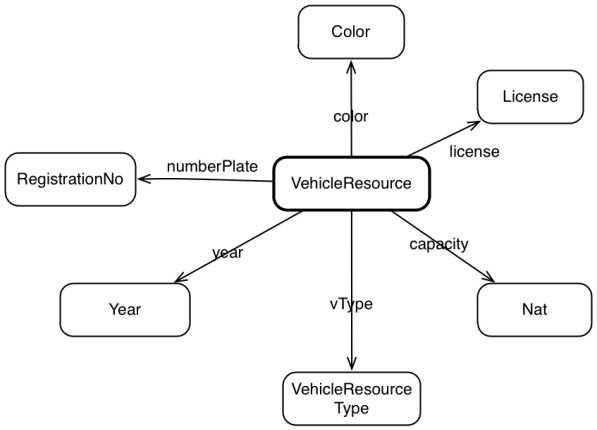


Figure 33.5: Local structural diagram of *VehicleResource* blob.

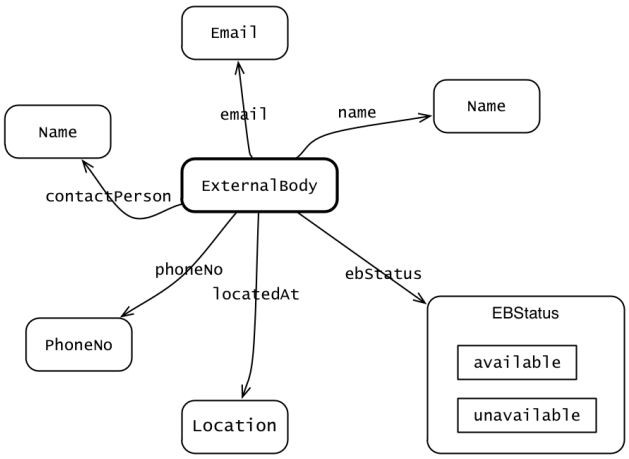


Figure 33.6: Local structural diagram of *ExternalBody* blob.

33.1.5 SkillType

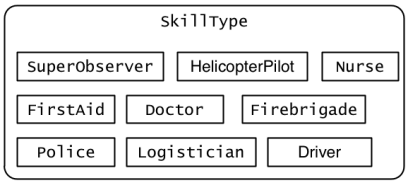


Figure 33.7: Local structural diagram of *SkillType* blob.

33.1.6 VehicleResourceType

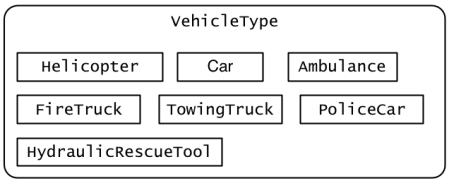


Figure 33.8: Local structural diagram of *VehicleResourceType* blob.

33.1.7 ResourceItem

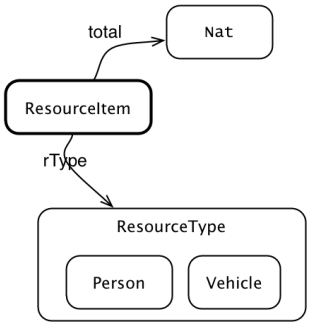


Figure 33.9: Local structural diagram of *ResourceItem* blob.

33.2 Behaviour

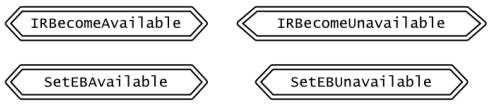


Figure 33.10: Global behavioural diagram of *Resources* package.

33.2.1 PersonResource

Figure 33.11: Local behavioural diagram of *PersonResource* blob.

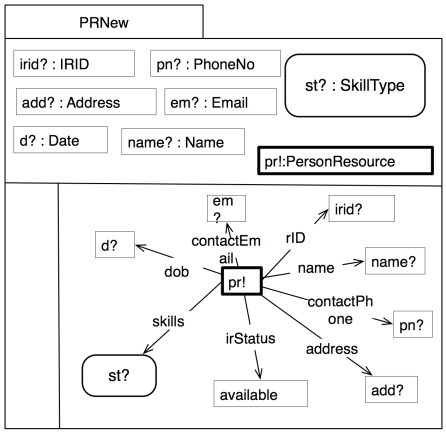


Figure 33.12: Contract diagram of *PersonResource*'s *PRNew* operation.

33.2.2 VehicleResource

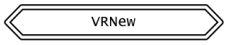


Figure 33.13: Local behavioural diagram of *VehicleResource* blob.

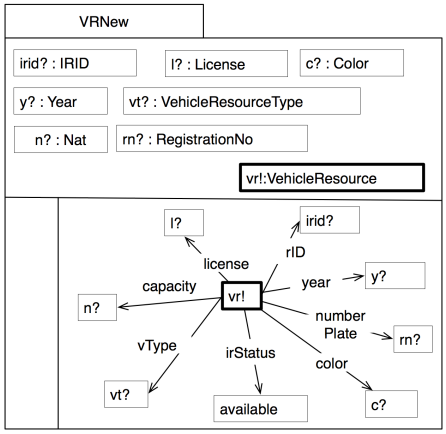


Figure 33.14: Contract diagram of *VehicleResource*'s *VRNew* operation.

33.2.3 ExternalBody

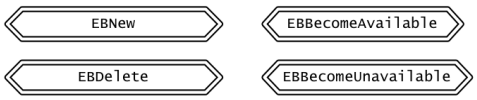


Figure 33.15: Local behavioural diagram of *ExternalBody* blob.

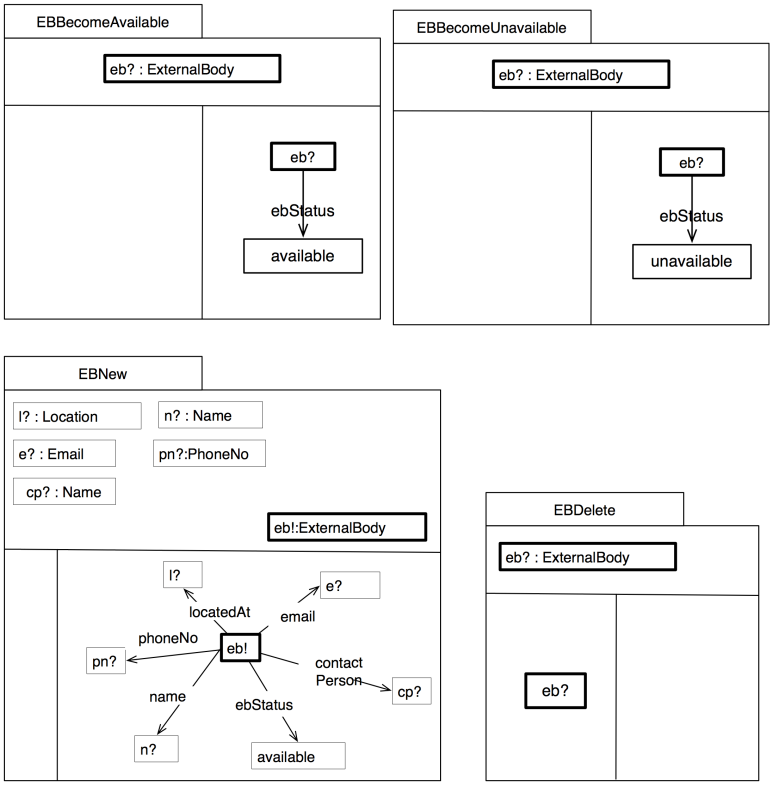


Figure 33.16: Contract diagrams of *ExternalBody* blob.

33.2.4 InternalResource

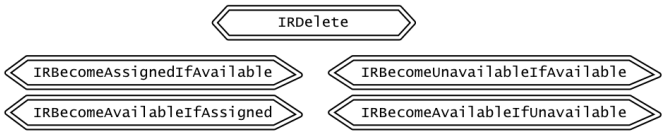


Figure 33.17: Local behavioural diagram of *InternalResource* blob.

33.2.5 Global Behaviour

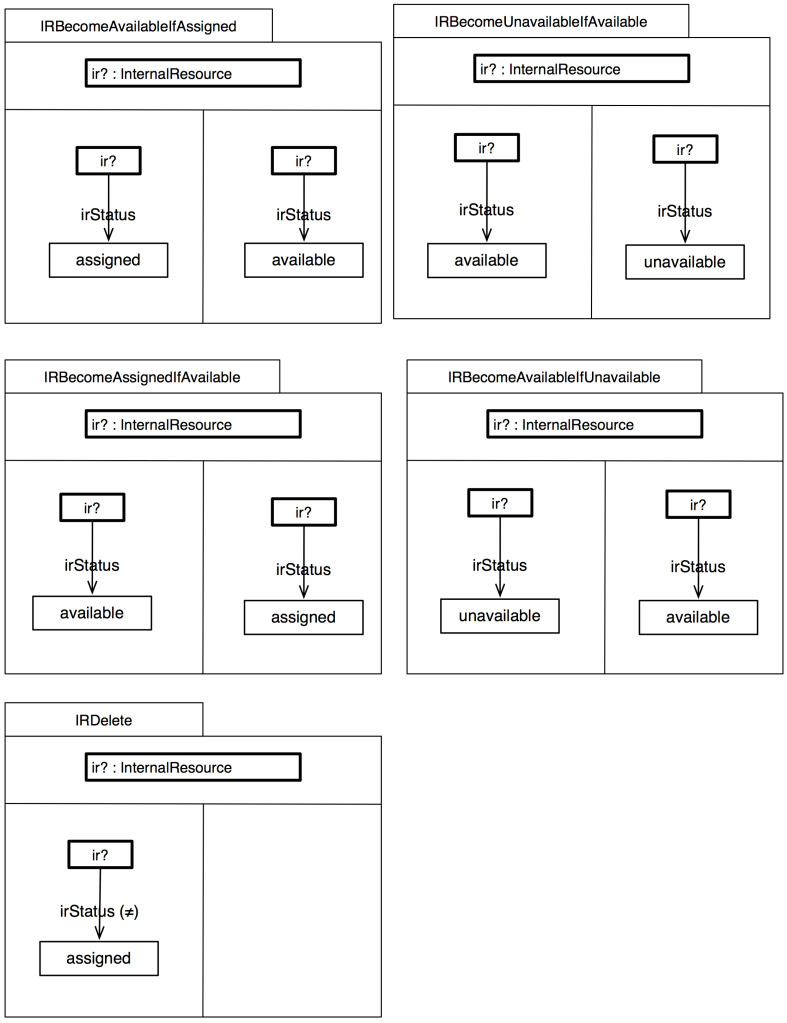


Figure 33.18: Contract diagrams of *InternalResource* blob.

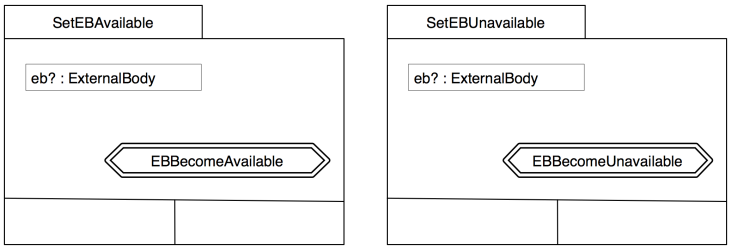


Figure 33.19: Contract diagrams of global operations *SetEBAvailable* and *SetEBUnavailable*.

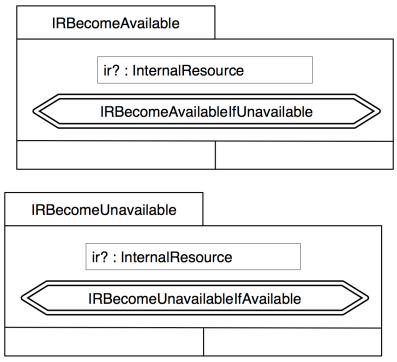


Figure 33.20: Contract diagrams of global operations *IRBecomeAvailable* and *IRBecomeUnavailable*.

Chapter 34

Package *Mission*

This chapter introduces package *Mission*, which addresses the concerns related with management of missions in CCCMS. Package *Mission* extends package *Resource* (chapter 33).

34.1 Structure

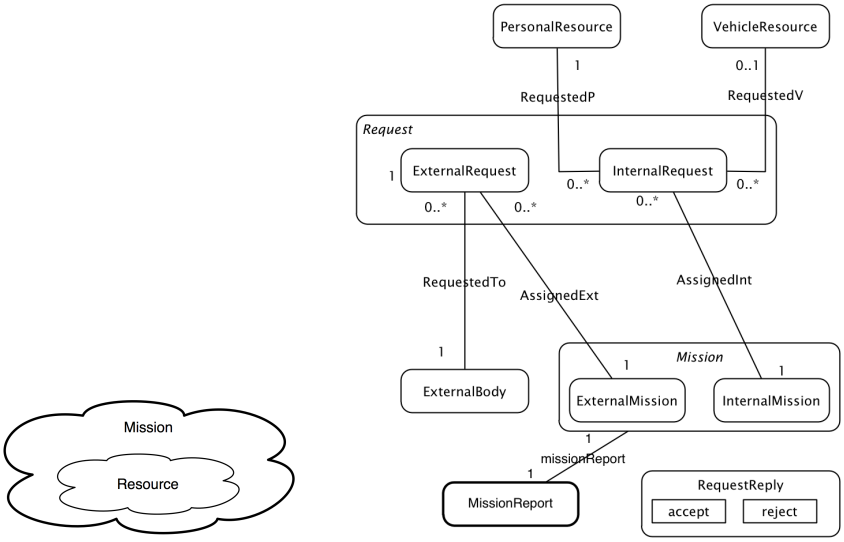


Figure 34.1: Pakcage *Mission* (left). Global structural diagram of *Mission* package (right).

34.1.1 Mission

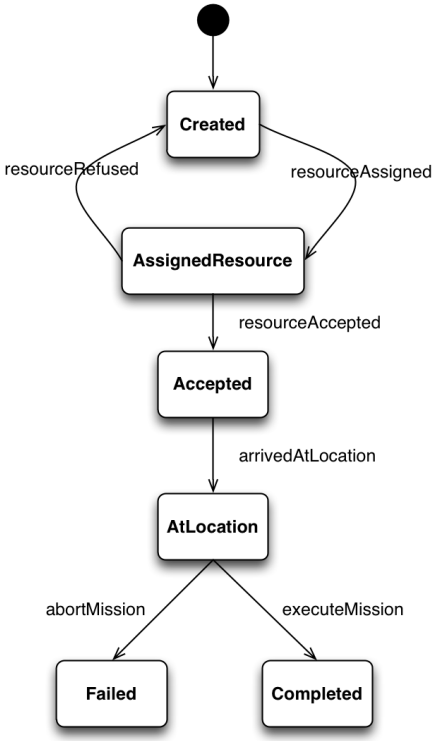


Figure 34.2: State diagram of the *Mission* blob.

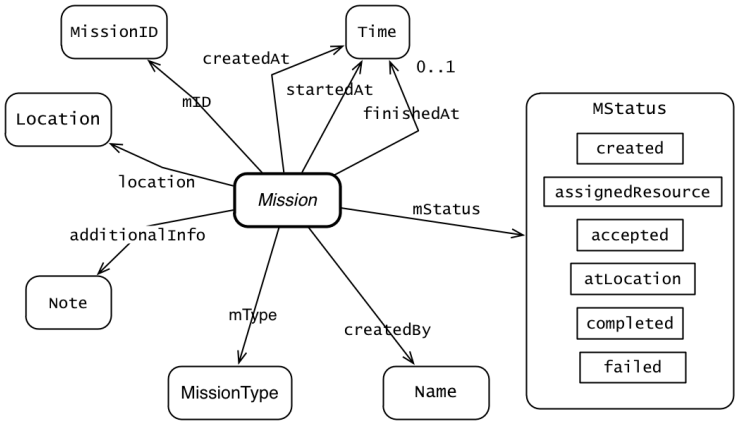


Figure 34.3: Local structural diagram of *Mission* blob.

34.1.2 MissionType

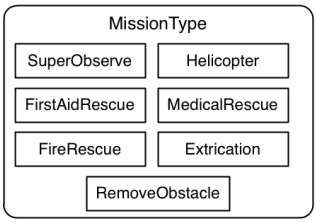


Figure 34.4: Local structural diagram of *MissionType* blob.

34.1.3 MissionReport

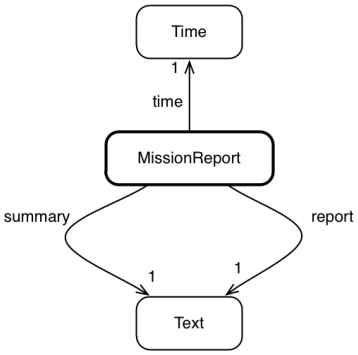


Figure 34.5: Local structural diagram of *MissionReport* blob.

34.1.4 Request

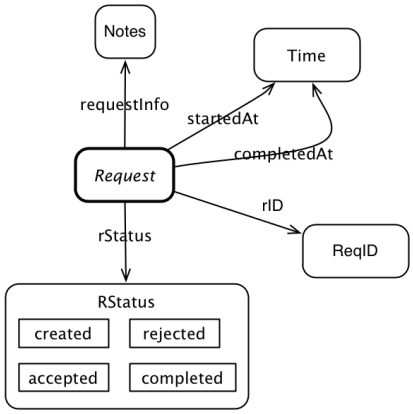


Figure 34.6: Local structural diagram of *Request* blob.

34.1.5 InternalRequest

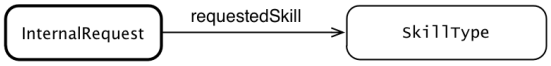


Figure 34.7: Local structural diagram of *InternalRequest* blob.

34.1.6 ExternalRequest

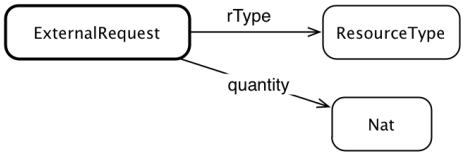


Figure 34.8: Local structural diagram of *ExternalRequest* blob.

34.2 Behaviour

34.2.1 Mission

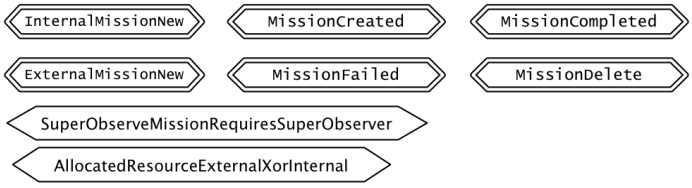


Figure 34.9: Local behavioural diagram of *Mission* blob.

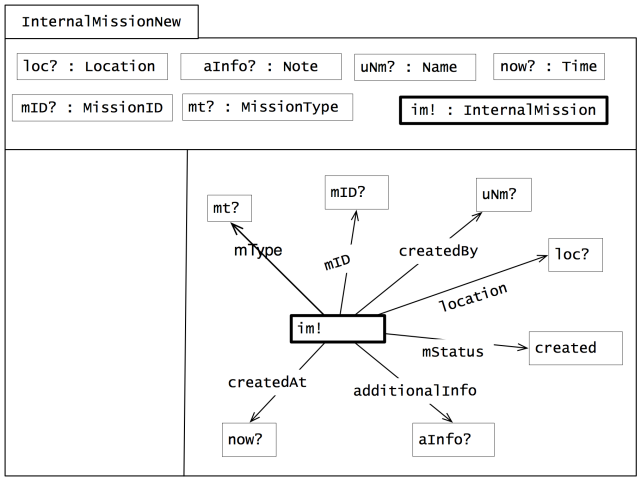


Figure 34.10: Local contract diagram for *InternalMissionNew* operation of *Mission* blob.

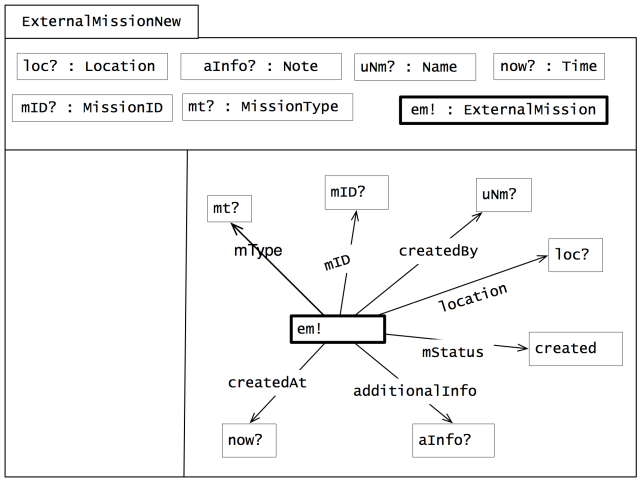


Figure 34.11: Local contract diagram for *ExternalMissionNew* operation of *Mission* blob.

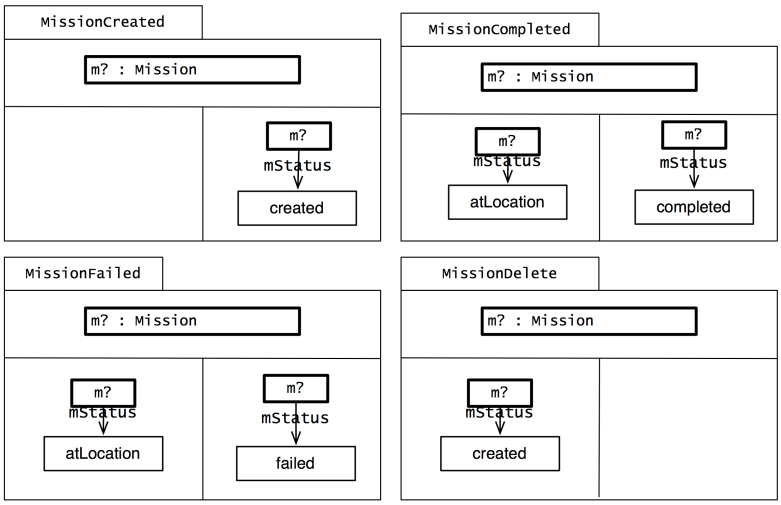


Figure 34.12: Local contract diagrams for mission state related operations of *Mission* blob.

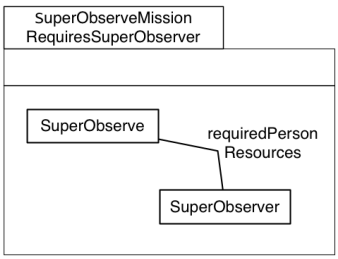


Figure 34.13: Constraint diagram of *Mission*'s *SuperObserveMissionRequiresSuperObserver* constraint.

34.2.2 MissionReport

Figure 34.14: Local behavioural diagram of *MissionReport* blob.

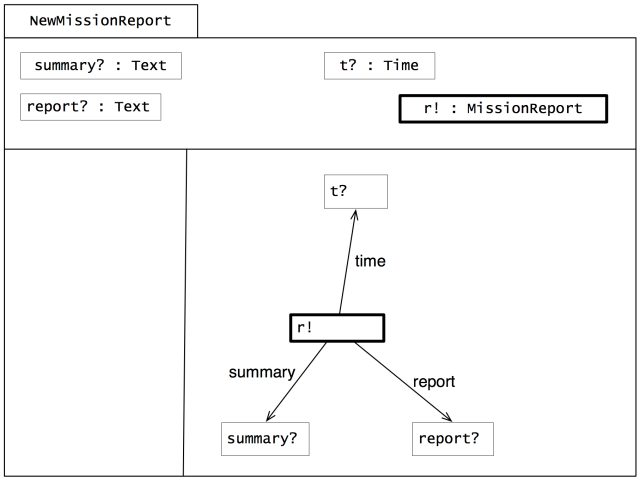


Figure 34.15: Local contract diagram for *NewMissionReport* operation of *MissionReport* blob.

34.2.3 Request

Figure 34.16: Local behavioural diagram of *Request* blob.

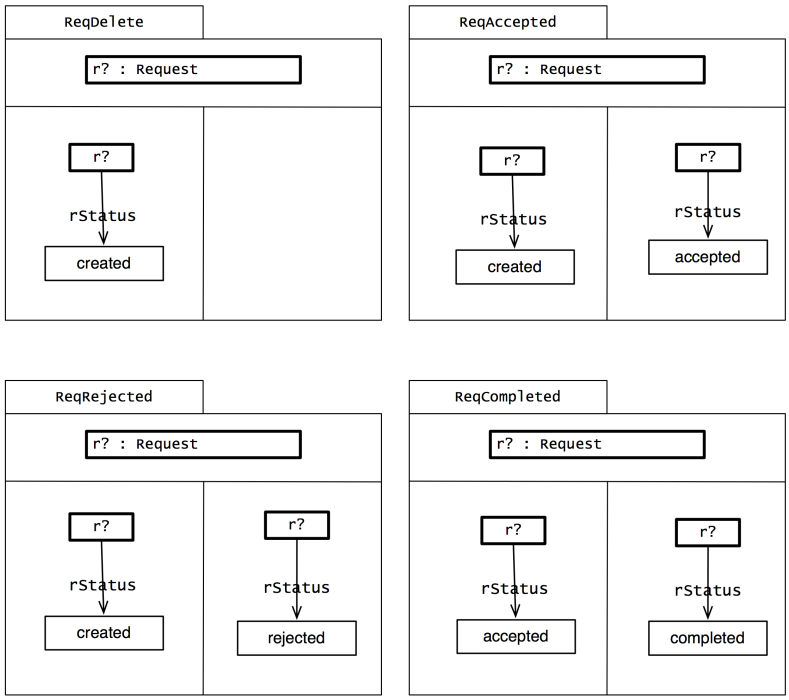


Figure 34.17: Local contract diagrams for operations of the *Request* blob.

34.2.4 InternalRequest

Figure 34.18: Local behavioural diagram of *InternalRequest* blob.

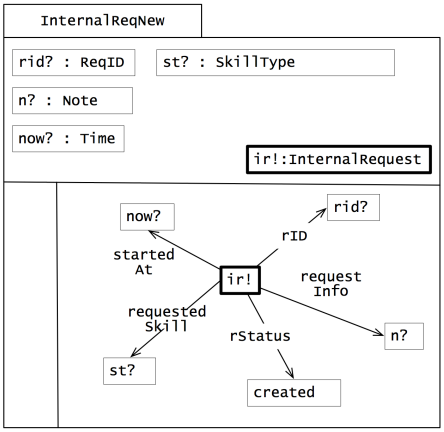


Figure 34.19: Local contract diagram for *InternalReqNew* operation of *InternalRequest* blob.

34.2.5 ExternalRequest

Figure 34.20: Local behavioural diagram of *ExternalRequest* blob.

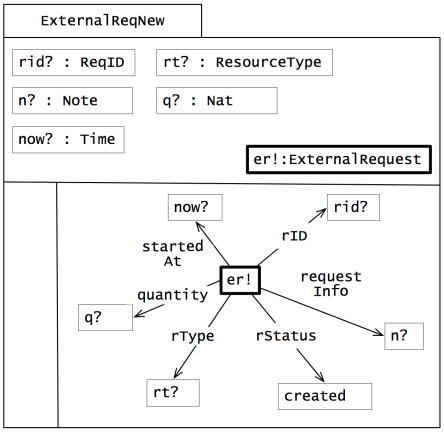


Figure 34.21: Local contract diagram for *ExternalReqNew* operation of *ExternalRequest* blob.

34.2.6 Global Behaviour

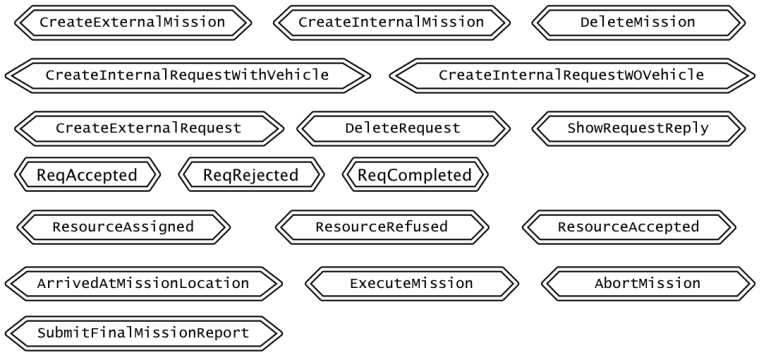


Figure 34.22: Global behavioural diagram of *Mission* package.

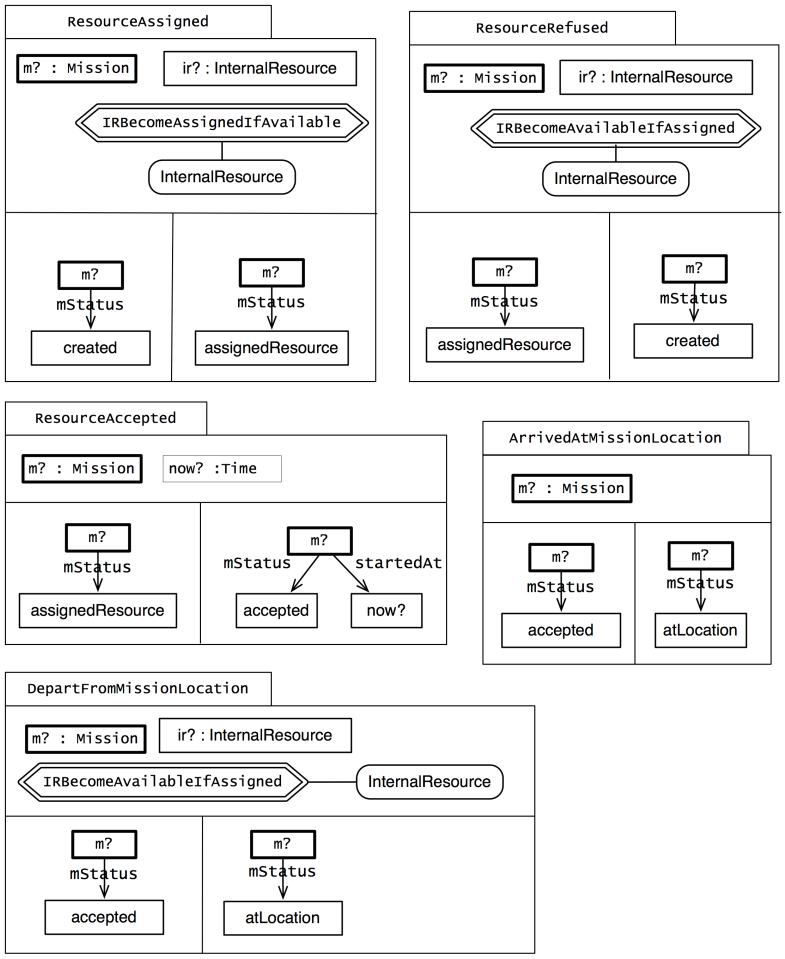


Figure 34.23: Contract diagram for resource-related global operations of the *Mission* package.

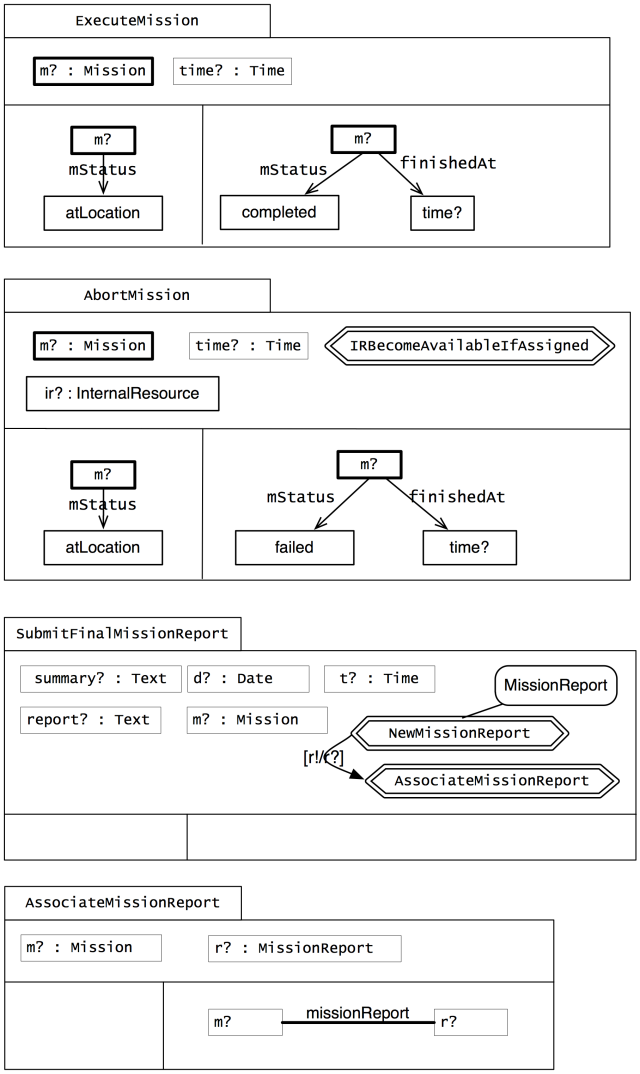


Figure 34.24: Contract diagram for Mission-related and MissionReport-related global operations of the *Mission* package.

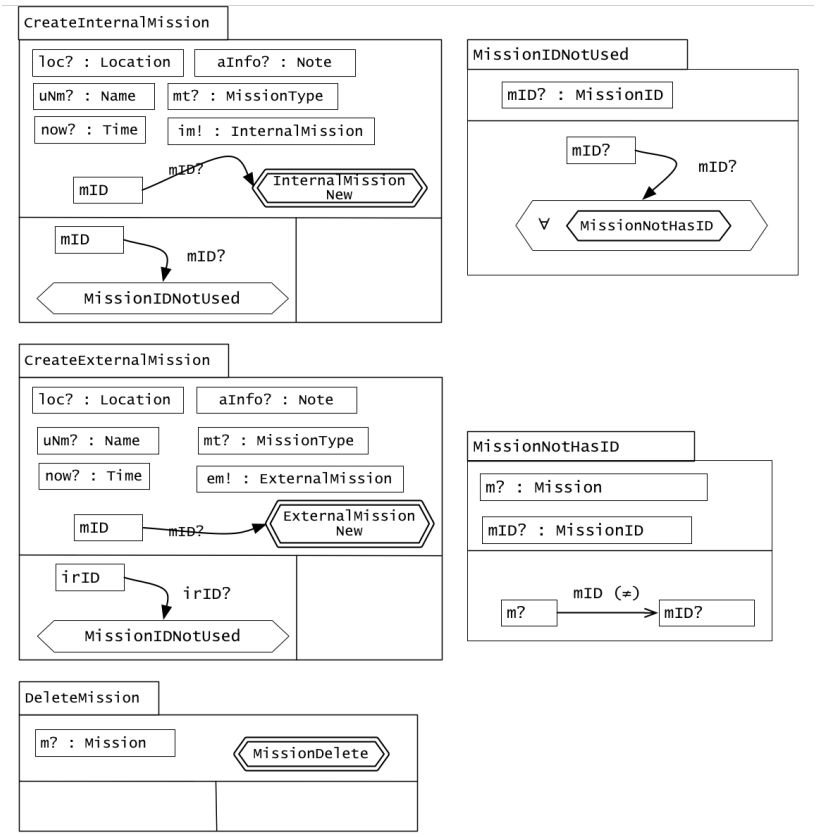


Figure 34.25: Contract diagrams for global mission-related operations of *Mission* package.

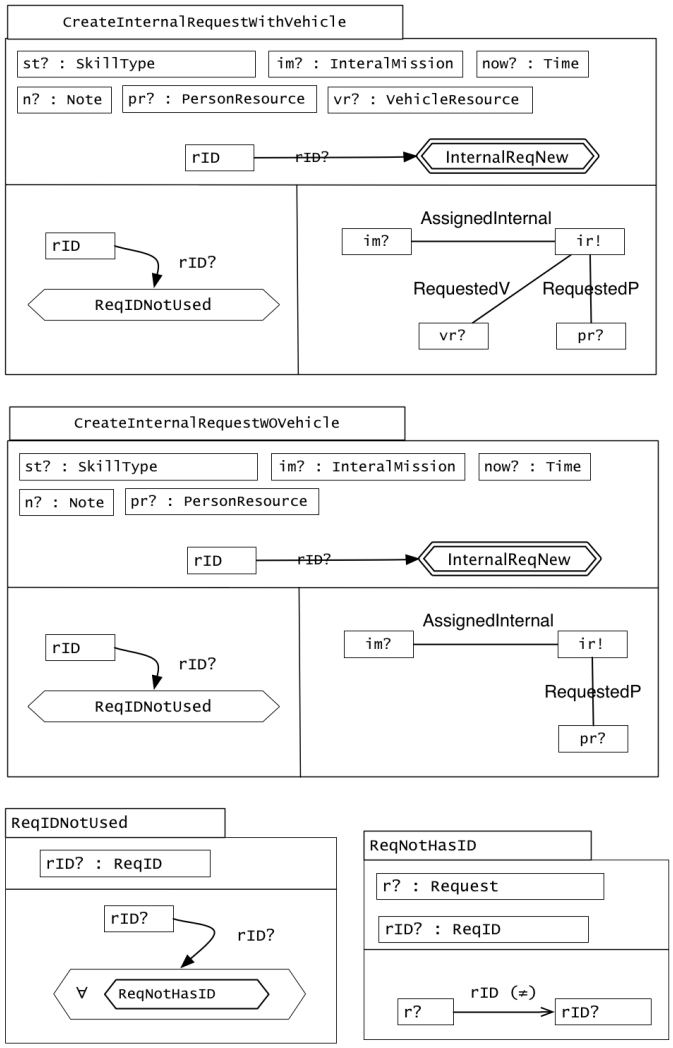


Figure 34.26: Contract diagrams for global *InternalRequest*-related operations of *Mission* package.

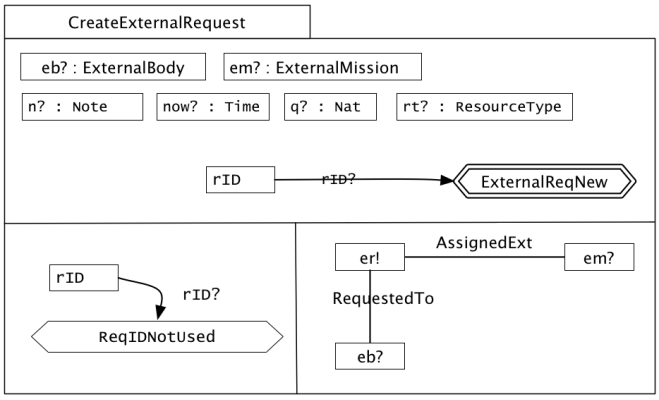


Figure 34.27: Contract diagram for global *CreateExternalRequest* operation of *Mission* package.

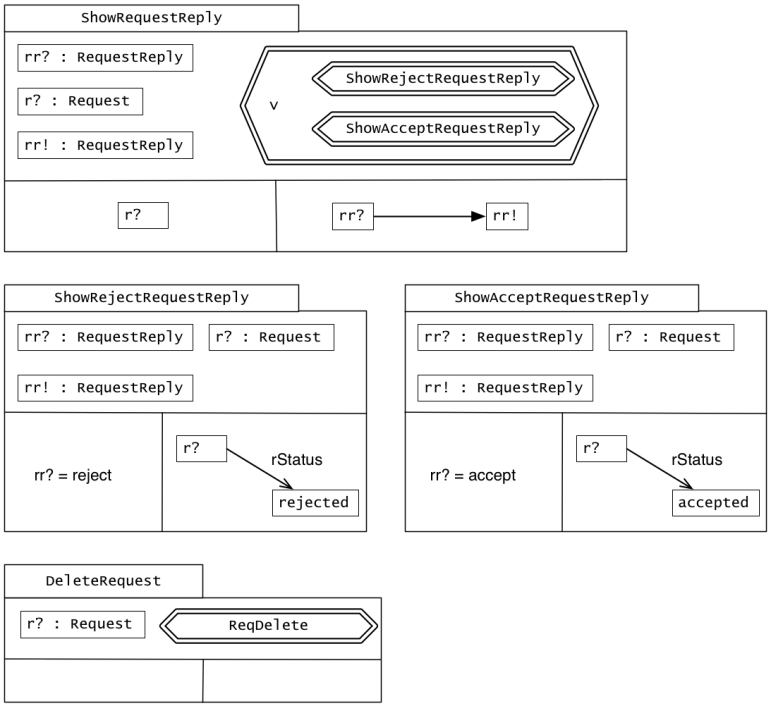


Figure 34.28: Contract diagrams for global *RequestReply*-related operations and for *DeleteRequest* operation of *Mission* package.

Chapter 35

Package *MappingDisplayMobCCCMS*

This chapter introduces package *MappingDisplayMobCCCMS*, which defines the mapping interface for *MobCCCMS* sub-system. Package *MappingDisplayMobCCCMS* extends package *MappingDisplayWithMapSys* (chapter 29).

35.1 Structure

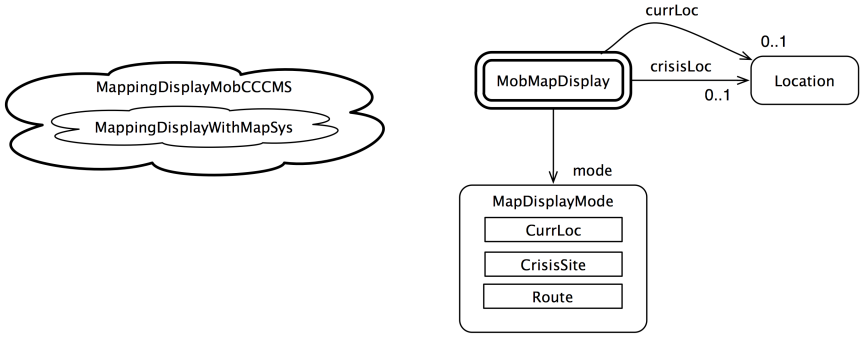


Figure 35.1: Package (left) and structural diagram of package *MappingDisplayMobCCCMS* (right).

35.2 Behaviour

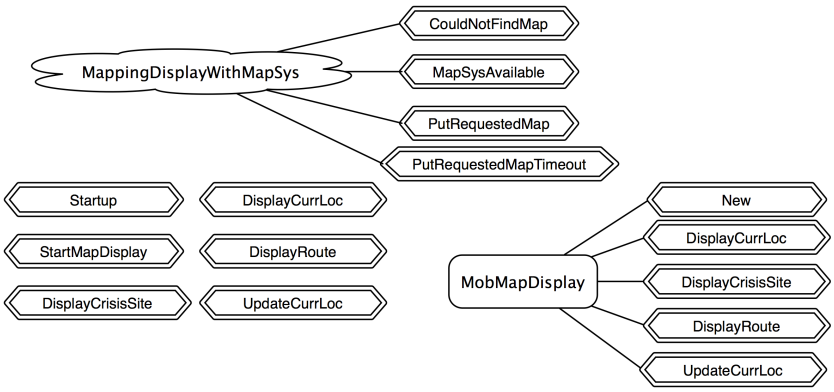


Figure 35.2: Behaviour diagram of package *MappingDisplayMobCCCMS*.

35.2.1 Blob *MobMapDisplay*

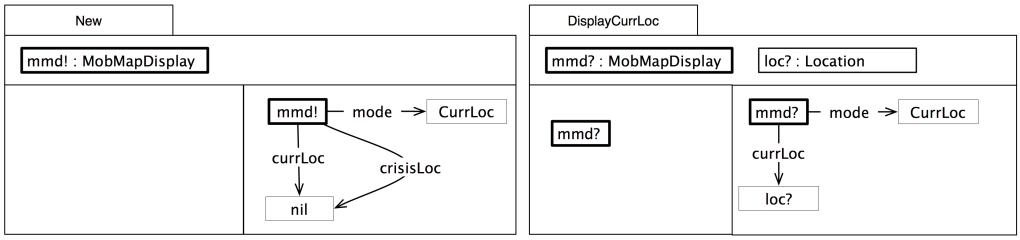


Figure 35.3: Contract diagrams for local operations *New* and *DisplayCurrLoc* of blob *MobMapDisplay*.

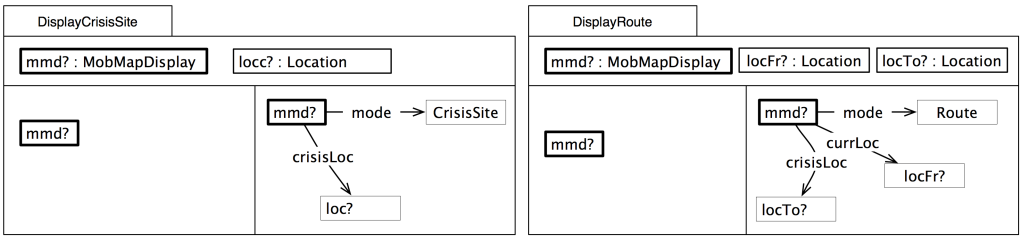


Figure 35.4: Contract diagrams for local operations *DisplayCrisisSite* and *DisplayRoute* of blob *MobMapDisplay*.

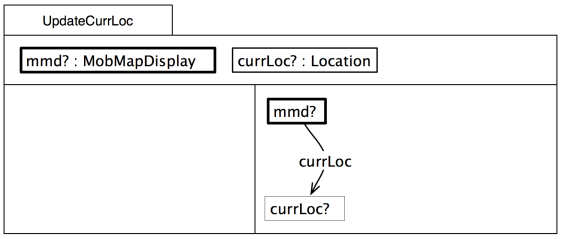


Figure 35.5: Contract diagram for local operations *UpdateCurrLoc* of blob *MobMapDisplay*.

35.2.2 Global Behaviour

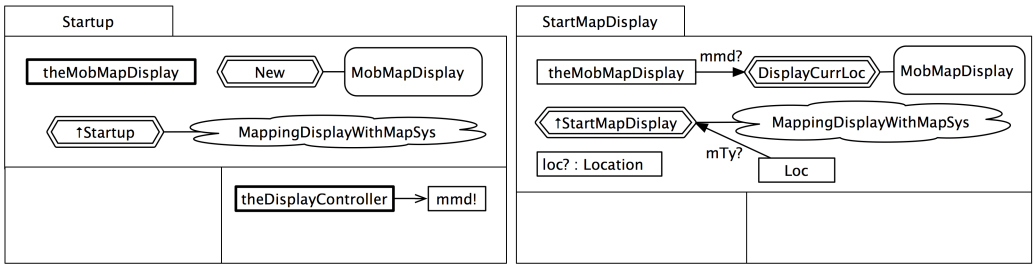


Figure 35.6: Contract diagrams for global operations *Startup* and *StartMapDisplay*.

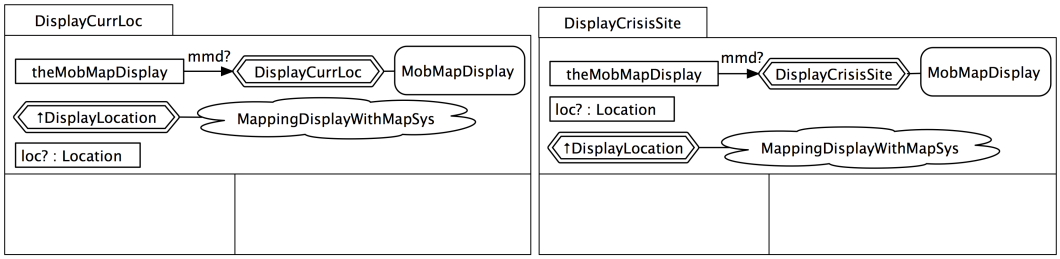


Figure 35.7: Contract diagrams for global operations *DisplayCurrLoc* and *DisplayCrisisSite*.

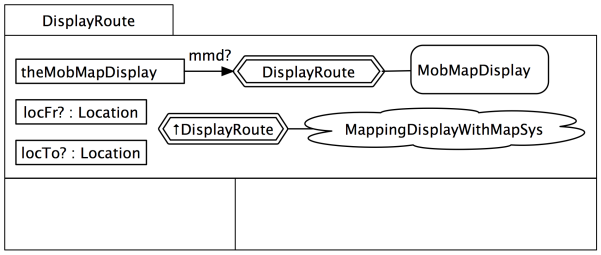


Figure 35.8: Contract diagrams for global operations *DisplayRoute*.

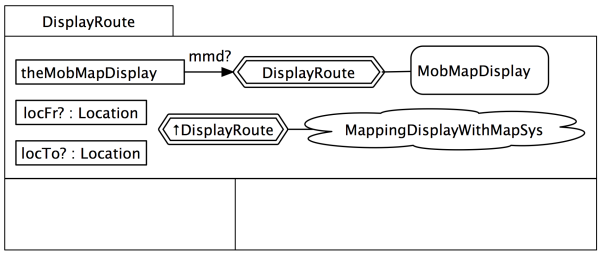


Figure 35.9: Contract diagram for global operations *DisplayRoute*.

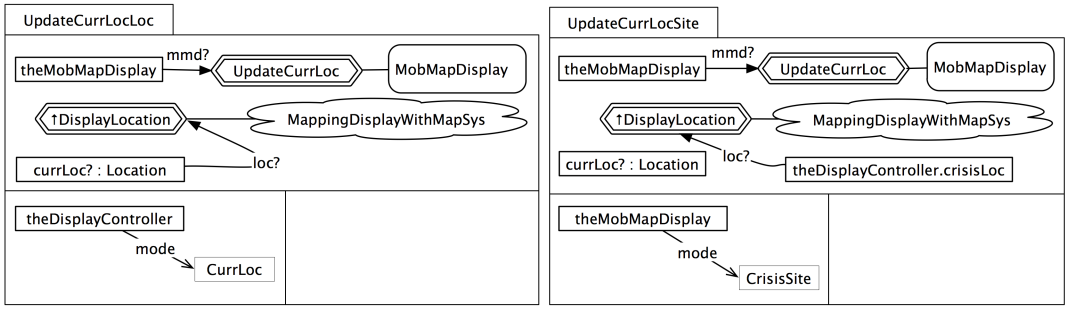


Figure 35.10: Contract diagrams for global operations *UpdateCurrLocLoc* and *UpdateCurrLocSite* that are used to form global operation *UpdateCurrLoc*.

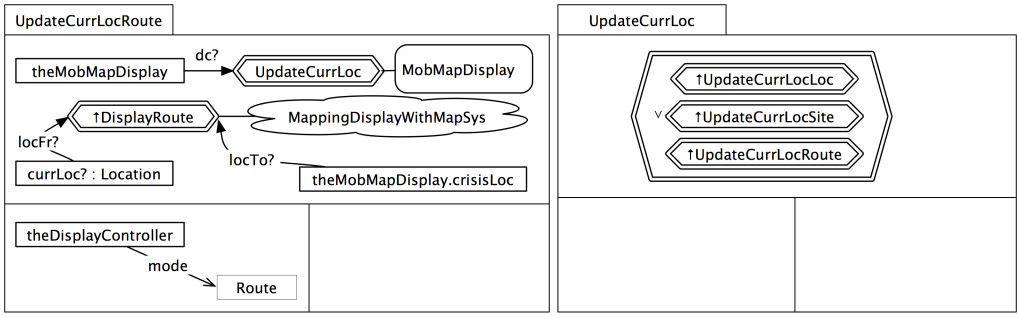


Figure 35.11: Contract diagrams for global operations *UpdateCurrLocRoute* and *UpdateCurrLoc*; *UpdCurrLoc* is disjunction of operations *UpdateCurrLocLoc*, *UpdateCurrLocSite* and *UpdateCurrLocRoute*.

Part VI

Joining Aspects into Packages

Chapter 36

Package *CrisisWithVS*

This chapter introduces package *CrisisWithVS* (*Crisis With Video Surveillance*), which adds the video surveillance concern to domain package *Crisis* (chapter 32). Package *CrisisWithVS* extends packages *Crisis* (chapter 32) and *VideoSurveillanceCCCMS* (chapter 27).

36.1 Structure

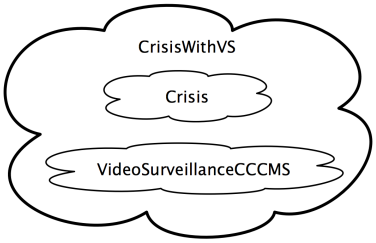


Figure 36.1: Package Diagram of package *CrisisWithVS*, which extends packages *Crisis* and *VideoSurveillanceCCCMS*.

36.2 Behaviour

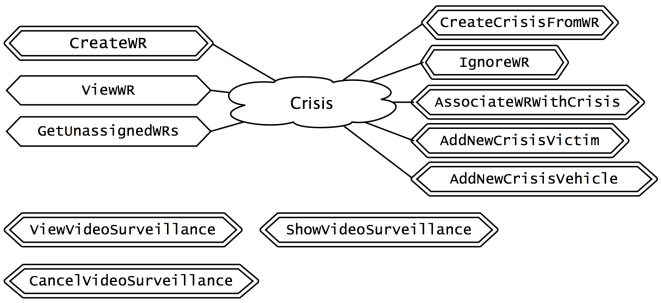


Figure 36.2: Behavioural diagram of package *CrisisWithVS*. Most operations are defined using *integral-extension*. Operations *ViewVideoSurveillance*, *ShowVideoSurveillance* and *CancelVideoSurveillance* are defined using custom extension in their own contract diagrams.

36.2.1 Global Operations

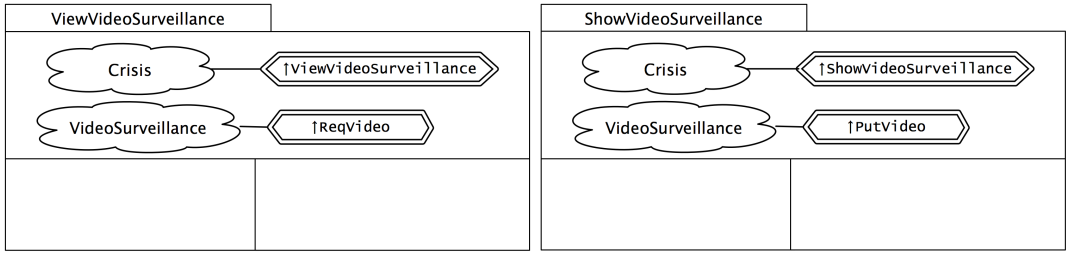


Figure 36.3: Contract diagrams of global operations *ViewVideoSurveillance* and *ShowVideoSurveillance*.

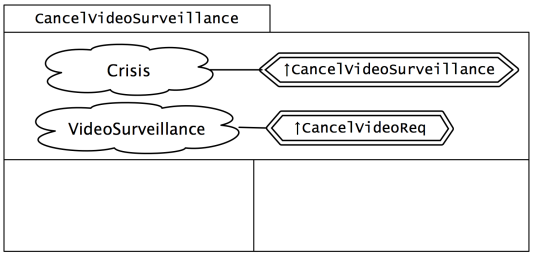


Figure 36.4: Contract diagrams of global operation *CancelVideoSurveillance*.

Chapter 37

Package *CrisisLoggingJI*

This chapter introduces package *CrisisLoggingJI* (*Crisis Logging Join Interface*), which defines the interface of package *Crisis* (chapter 32) for the logging aspect. A generic package for logging is defined in chapter 15; its customisation package (*LoggingCCCMS*) is defined in chapter 24. Package *CrisisLoggingJI* sees package *EventsCCCMS* (chapter 23); the interface consists of adding an output of *EventKind* (defined in the *EventsCCCMS* package), which says which logging event corresponds to which global operation.

37.1 Structure

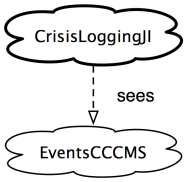


Figure 37.1: Package *CrisisLoggingJI*.

37.2 Behaviour

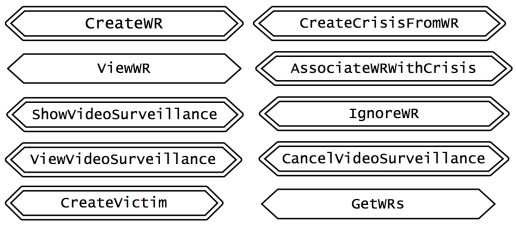


Figure 37.2: Global behavioural diagram of package *CrisisLoggingJI*.

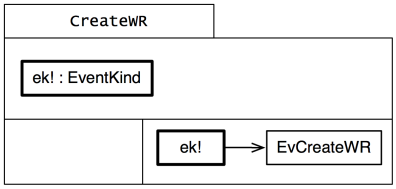


Figure 37.3: Extension of contract *CreateWR* with an *EventKind* output.

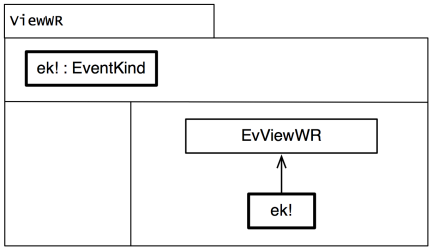


Figure 37.4: Extension of contract *ViewWR* with an *EventKind* output.

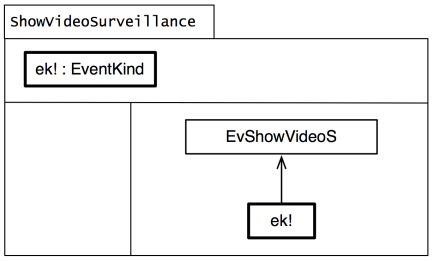


Figure 37.5: Extension of contract *ShowVideoSurveillance* with an *EventKind* output.

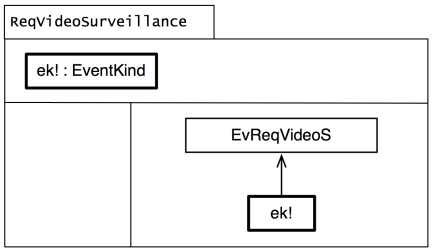


Figure 37.6: Extension of contract *ReqVideoSurveillance* with an *EventKind* output.

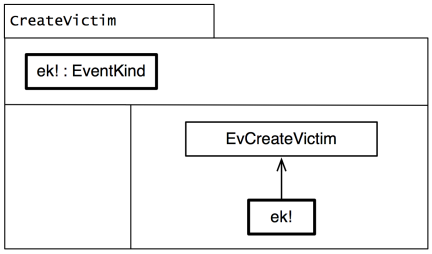


Figure 37.7: Extension of contract *CreateVictim* with an *EventKind* output.

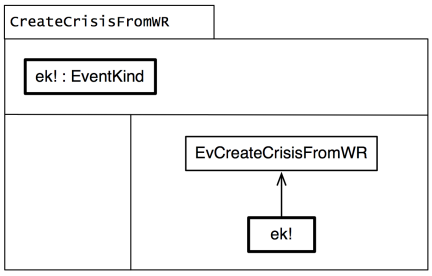


Figure 37.8: Extension of contract *CreateCrisisFromWR* with an *EventKind* output.

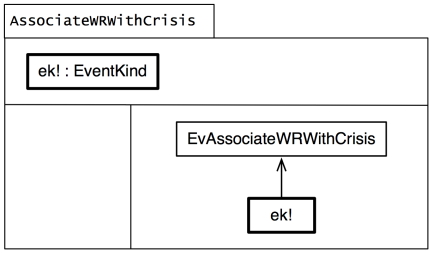


Figure 37.9: Extension of contract *AssociateWRWithCrisis* with an *EventKind* output.

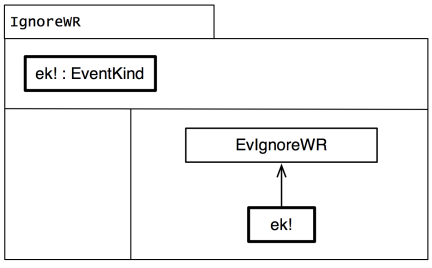


Figure 37.10: Extension of contract *IgnoreWR* with an *EventKind* output.

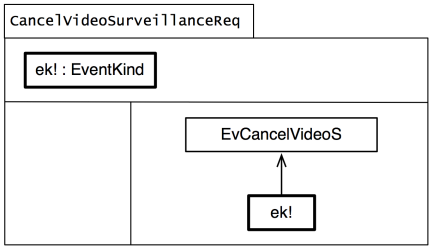


Figure 37.11: Extension of contract *CancelVideoSurveillanceReq* with an *EventKind* output.

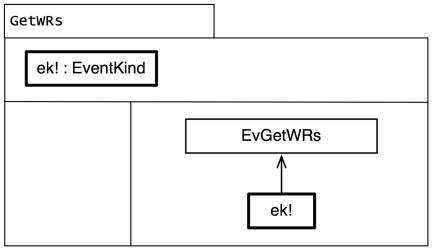


Figure 37.12: Extension of contract *GetWRs* with an *EventKind* output.

Chapter 38

Package *MissionLoggingJI*

This chapter introduces package *MissionLoggingJI* (*Mission Logging Join Interface*), which defines the interface of package *Mission* (chapter 34) for the logging aspect. A generic package for logging is defined in chapter 15; its customisation package (*LoggingCCCMS*) is defined in chapter 24. Package *MissionLoggingJI* sees package *EventsCCCMS* (chapter 23); the interface consists of adding an output of *EventKind* (defined in the *EventsCCCMS* package), which says which logging event corresponds to which global operation.

38.1 Structure

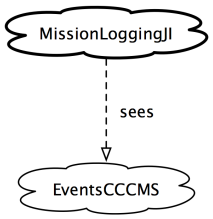


Figure 38.1: Package *MissionLoggingJI* sees package *EventsCCCMS*.

38.2 Behaviour

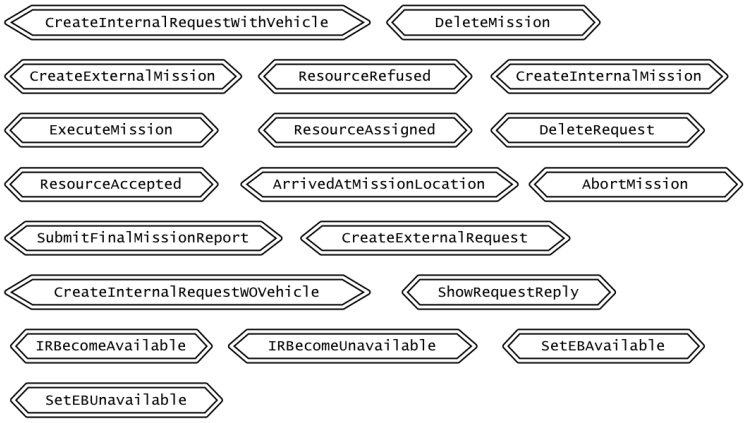


Figure 38.2: Global behavioural diagram of package *MissionLoggingJoinInterface*.

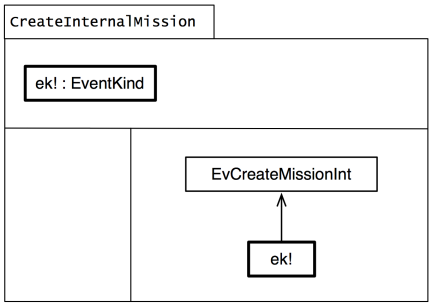


Figure 38.3: Extension of contract *CreateInternalMission* with an *EventKind* output.

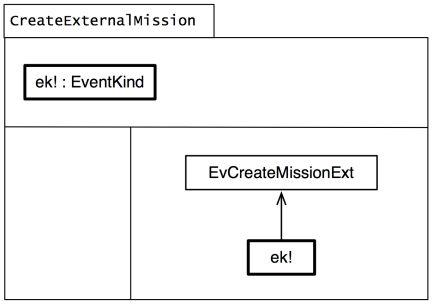


Figure 38.4: Extension of contract *CreateExternalMission* with an *EventKind* output.

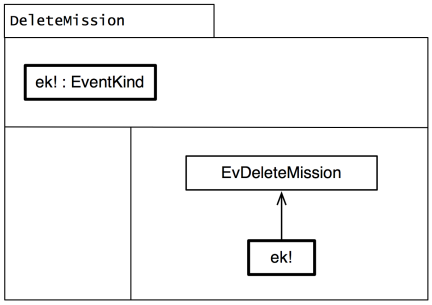


Figure 38.5: Extension of contract *DeleteMission* with an *EventKind* output.

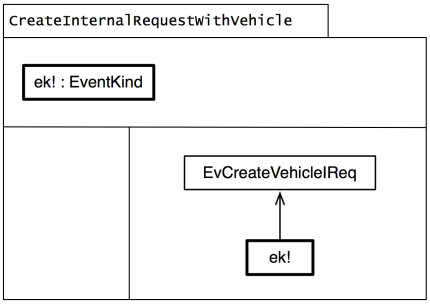


Figure 38.6: Extension of contract *CreateInternalRequestWithVehicle* with an *EventKind* output.

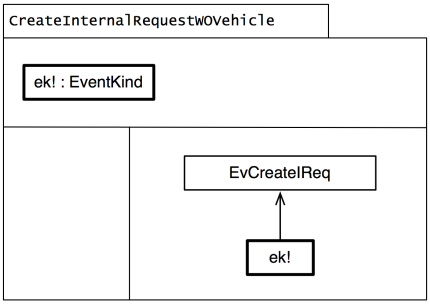


Figure 38.7: Extension of contract *CreateInternalRequestWOVehicle* with an *EventKind* output.

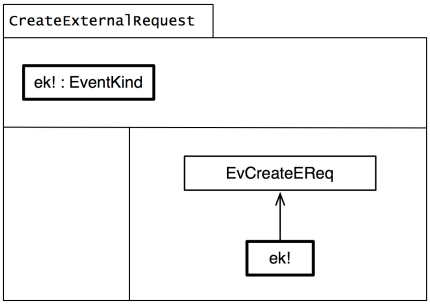


Figure 38.8: Extension of contract *CreateExternalRequest* with an *EventKind* output.

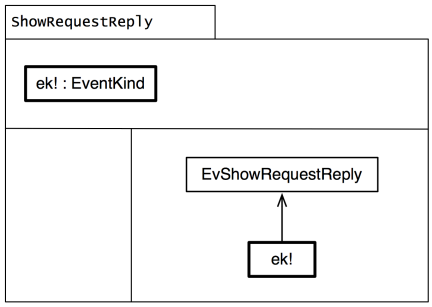


Figure 38.9: Extension of contract *ShowRequestReply* with an *EventKind* output.

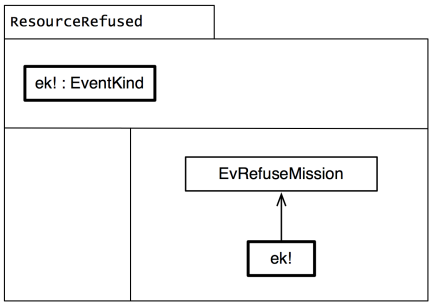


Figure 38.10: Extension of contract *ResourceRefused* with an *EventKind* output.

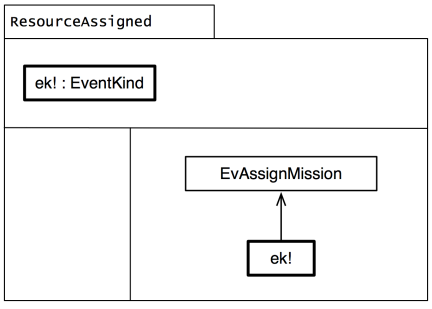


Figure 38.11: Extension of contract *ResourceAssigned* with an *EventKind* output.

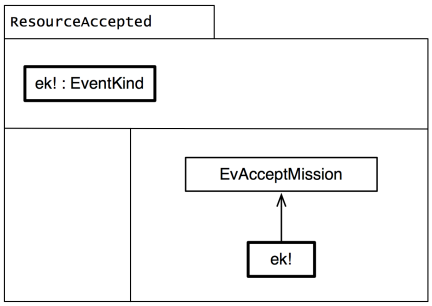


Figure 38.12: Extension of contract *ResourceAccepted* with an *EventKind* output.

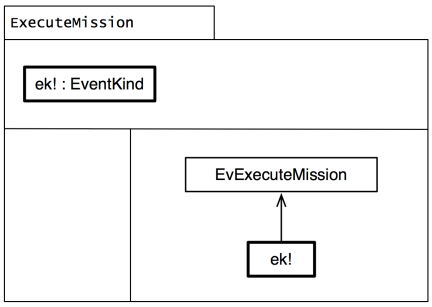


Figure 38.13: Extension of contract *ExecuteMission* with an *EventKind* output.

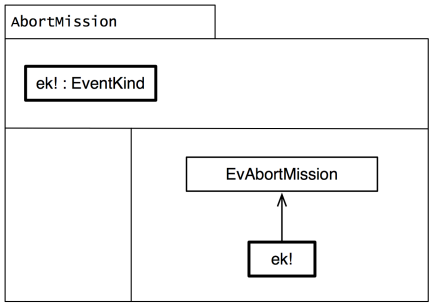


Figure 38.14: Extension of contract *AbortMission* with an *EventKind* output.

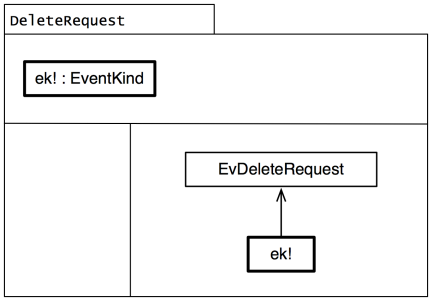


Figure 38.15: Extension of contract *DeleteRequest* with an *EventKind* output.

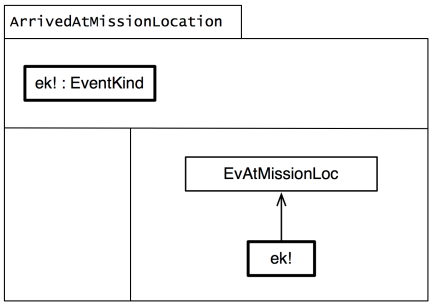


Figure 38.16: Extension of contract *ArrivedAtMissionLocation* with an *EventKind* output.

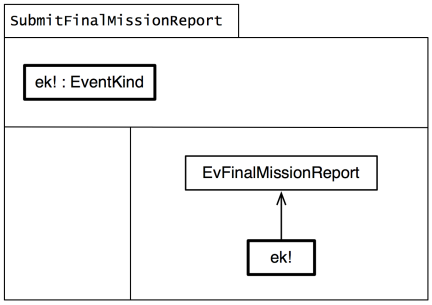


Figure 38.17: Extension of contract *SubmitFinalMissionReport* with an *EventKind* output.

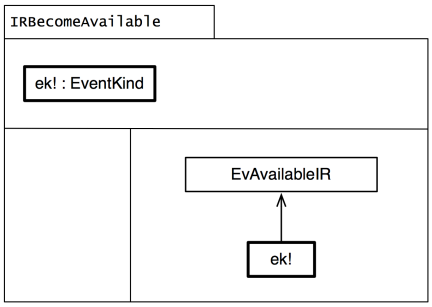


Figure 38.18: Extension of contract *IRBecomeAvailable* with an *EventKind* output.

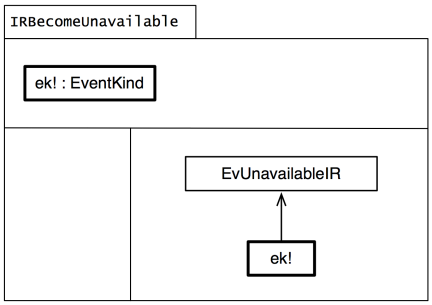


Figure 38.19: Extension of contract *IRBecomeUnavailable* with an *EventKind* output.

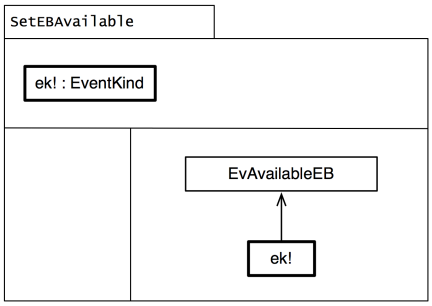


Figure 38.20: Extension of contract *SetEBAvailable* with an *EventKind* output.

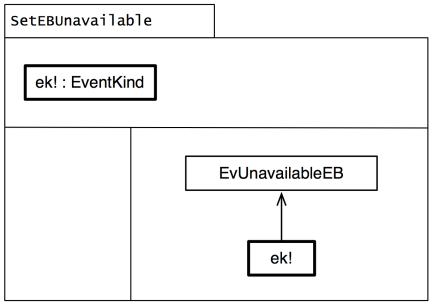


Figure 38.21: Extension of contract *SetEBUnavailable* with an *EventKind* output.

Chapter 39

Package *CrisisAuthorisationJI*

This chapter introduces package *CrisisAuthorisationJI* (*Crisis Authorisation Join Interface*), which defines the interface of package *Crisis* (chapter 32) for the authenticated access-control aspect. A generic package for authorisation is defined in chapter 12; its customisation package (*AuthorisationCCCMS*) is defined in chapter 22. Package *CrisisAuthorisationJI* sees package *ACTasksCCCMS* (chapter 21); the interface consists of adding an output of *Task* (defined in the *ACTasksCCCMS* package), which says which task corresponds to which global operation of package *Crisis*.

39.1 Structure

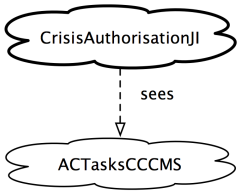


Figure 39.1: Package *CrisisAuthorisationJI*.

39.2 Behaviour

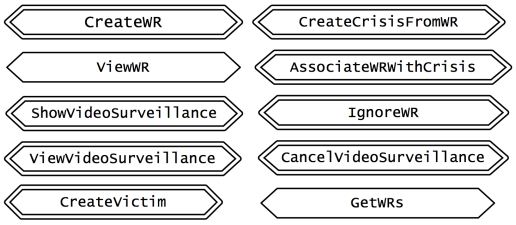


Figure 39.2: Global behavioural diagram of package *CrisisAuthorisationJoinInterface*.

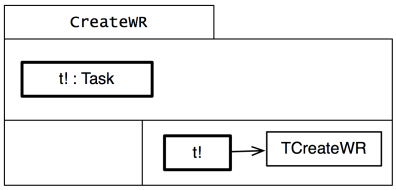


Figure 39.3: Extension of contract *CreateWR* with a *Task* output.

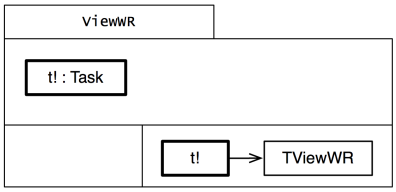


Figure 39.4: Extension of contract *ViewWR* with a *Task* output.

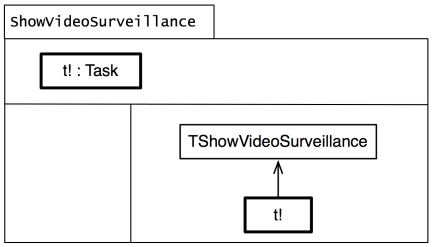


Figure 39.5: Extension of contract *ShowVideoSurveillance* with a *Task* output.

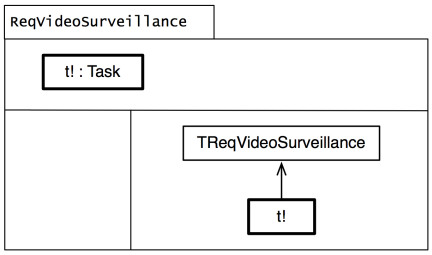


Figure 39.6: Extension of contract *ReqVideoSurveillance* with a *Task* output.

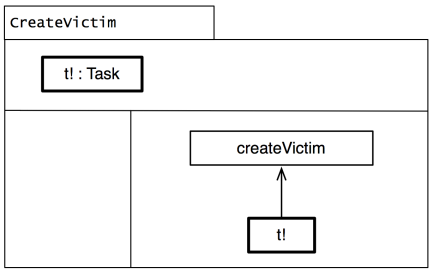


Figure 39.7: Extension of contract *CreateVictim* with a *Task* output.

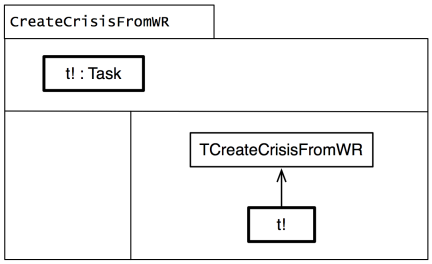


Figure 39.8: Extension of contract *CreateCrisisFromWR* with a *Task* output.

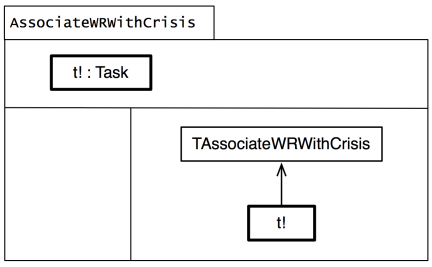


Figure 39.9: Extension of contract *AssociateWRWithCrisis* with a *Task* output.

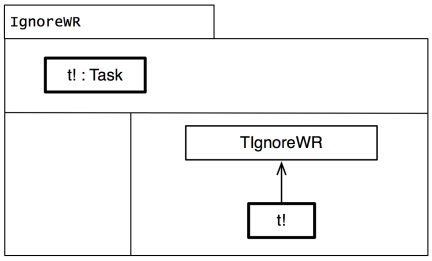


Figure 39.10: Extension of contract *IgnoreWR* with a *Task* output.

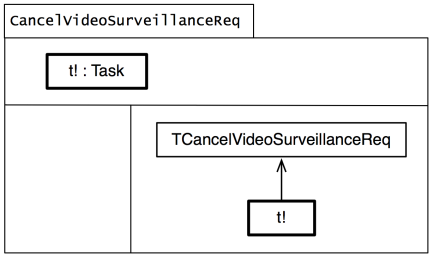


Figure 39.11: Extension of contract *CancelVideoSurveillanceReq* with a *Task* output.

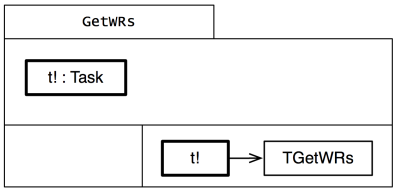


Figure 39.12: Extension of contract *GetWRs* with a *Task* output.

Chapter 40

Package *MissionAuthorisationJI*

This chapter introduces package *MissionAuthorisationJI* (*Mission Authorisation Join Interface*), which defines the interface of package *Mission* (chapter 34) for the authenticated access-control aspect. A generic package for authorisation is defined in chapter 12; its customisation package (*AuthorisationCCCMS*) is defined in chapter 22. Package *MissionAuthorisationJI* sees package *ACTasksCCCMS* (chapter 21); the interface consists of adding an output of *Task* (defined in the *ACTasksCCCMS* package), which says which task corresponds to which global operation of package *Mission*.

40.1 Structure

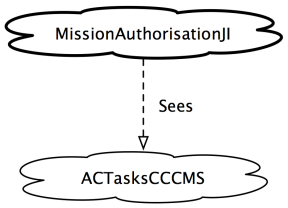


Figure 40.1: Package *MissionAuthorisationJI* sees package *ACTasksCCCMS*.

40.2 Behaviour

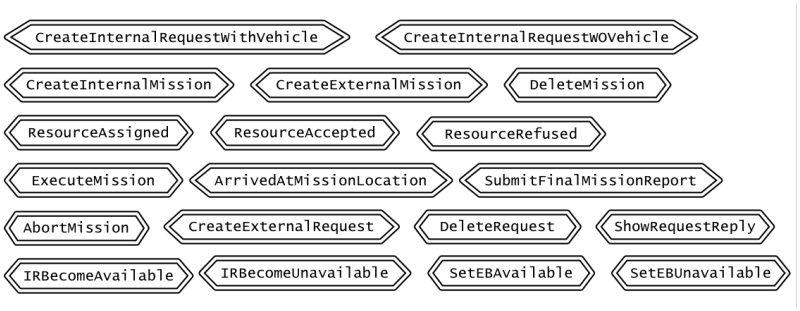


Figure 40.2: Global behavioural diagram of package *MissionAuthorisationJoinInterface*.

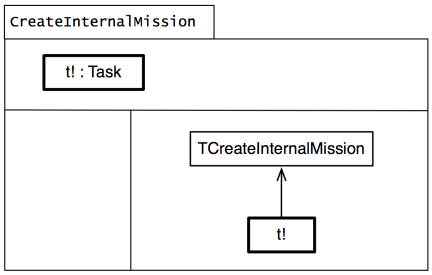


Figure 40.3: Extension of contract *CreateInternalMission* with a *Task* output.

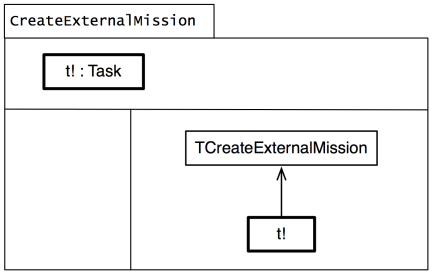


Figure 40.4: Extension of contract *CreateExternalMission* with a *Task* output.

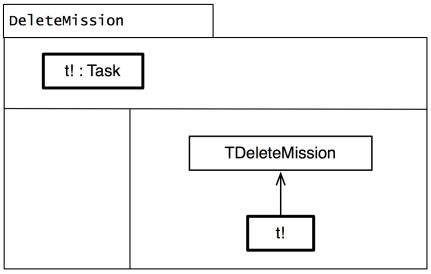


Figure 40.5: Extension of contract *DeleteMission* with a *Task* output.

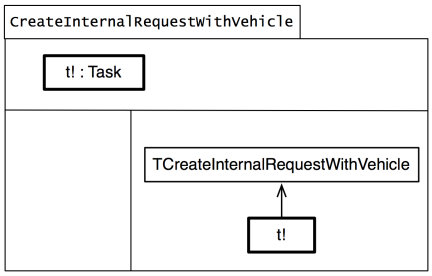


Figure 40.6: Extension of contract *CreateInternalRequestWithVehicle* with a *Task* output.

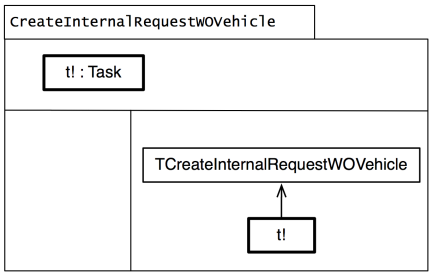


Figure 40.7: Extension of contract *CreateInternalRequestWOVehicle* with a *Task* output.

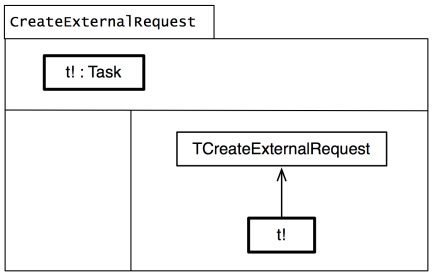


Figure 40.8: Extension of contract *CreateExternalRequest* with a *Task* output.

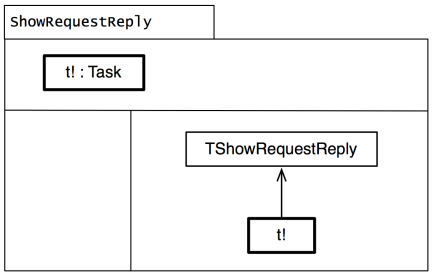


Figure 40.9: Extension of contract *ShowRequestReply* with a *Task* output.

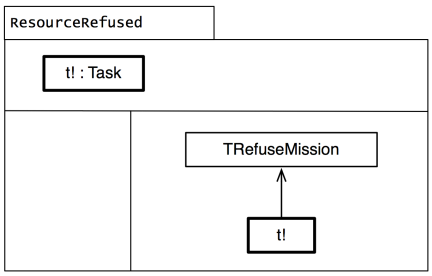


Figure 40.10: Extension of contract *ResourceRefused* with a *Task* output.

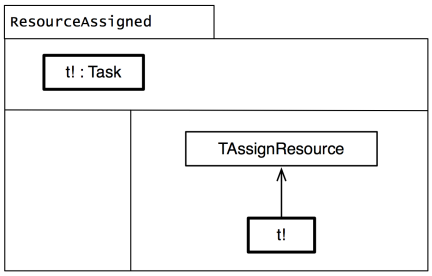


Figure 40.11: Extension of contract *ResourceAssigned* with a *Task* output.

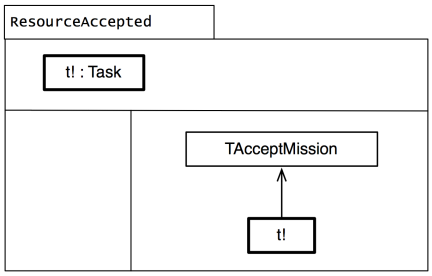


Figure 40.12: Extension of contract *ResourceAccepted* with a *Task* output.

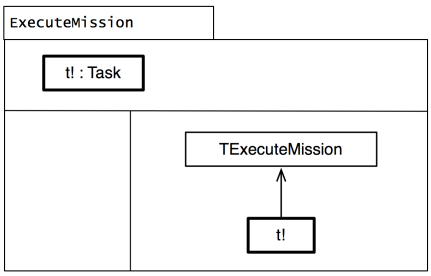


Figure 40.13: Extension of contract *ExecuteMission* with a *Task* output.

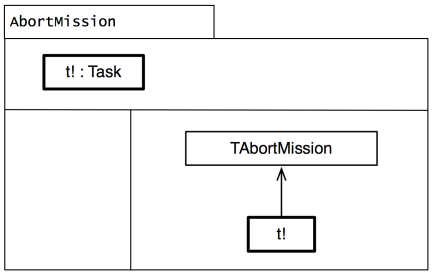


Figure 40.14: Extension of contract *AbortMission* with a *Task* output.

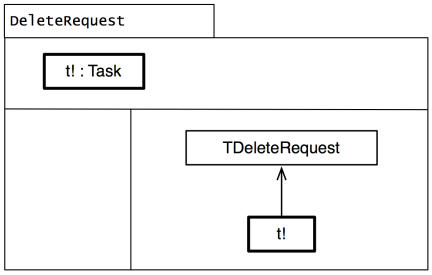


Figure 40.15: Extension of contract *DeleteRequest* with a *Task* output.

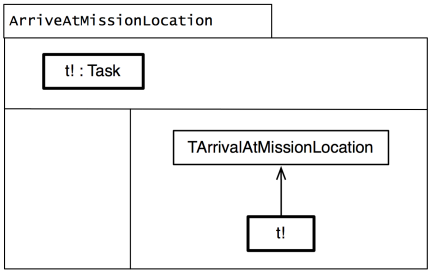


Figure 40.16: Extension of contract *ArrivedAtMissionLocation* with a *Task* output.

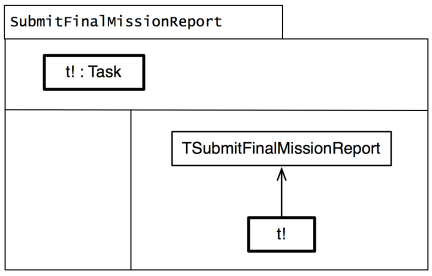


Figure 40.17: Extension of contract *SubmitFinalMissionReport* with a *Task* output.

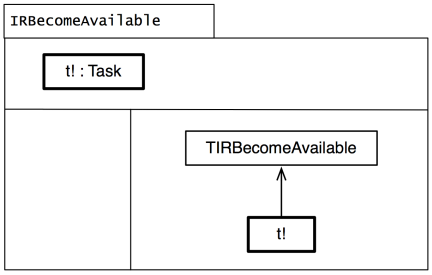


Figure 40.18: Extension of contract *IRBecomeAvailable* with a *Task* output.

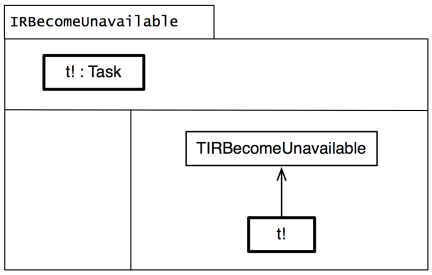


Figure 40.19: Extension of contract *IRBecomeUnavailable* with a *Task* output.

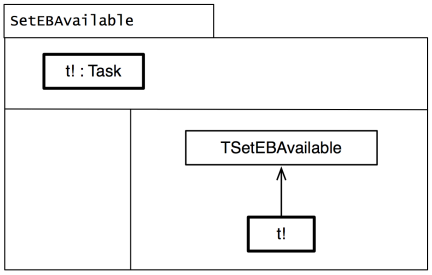


Figure 40.20: Extension of contract *SetEBAvailable* with a *Task* output.

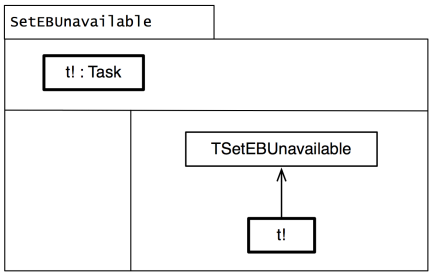


Figure 40.21: Extension of contract *SetEBUnavailable* with a *Task* output.

Chapter 41

Package *CrisisWithJI*

This chapter introduces package *CrisisWithJI* (*Crisis With Join Interfaces*), which add the required join interfaces to domain package *Crisis* (chapter 32). Essentially, this package augments the domain-only package *Crisis* with necessary interfaces to enable composition of required aspects. Package *CrisisWithJI* extends packages *CrisisWithVS* (chapter 36), *CrisisLoggingJI* (chapter 37) and *CrisisAuthorisationJI* (chapter 39).

41.1 Structure

Figure 41.1: Package *CrisisWithJI* extends packages *CrisisWithVS*, *CrisisLoggingJI* and *CrisisAuthorisationJI*.

41.2 Behaviour

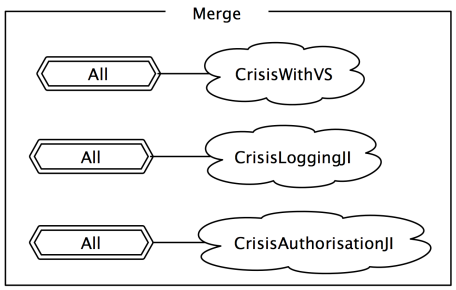


Figure 41.2: Behavioural diagram of package *CrisisWithJoinInterface*.

Chapter 42

Package *MissionWithJI*

This chapter introduces package *MissionWithJI* (*Mission With Join Interfaces*), which add the required join interfaces to domain package *Mission* (chapter 34). Essentially, this package augments the domain-only package *Mission* with necessary interfaces to enable composition of required aspects. Package *MissionWithJI* extends packages *Mission* (chapter 32), *MissionLoggingJI* (chapter 38) and *MissionAuthorisationJI* (chapter 40).

42.1 Structure

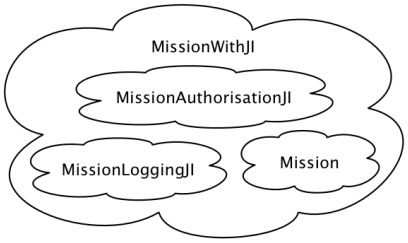


Figure 42.1: Package *MissionWithJI* extends packages *Mission*, *MissionLoggingJI* and *MissionAuthorisationJI*.

42.2 Behaviour

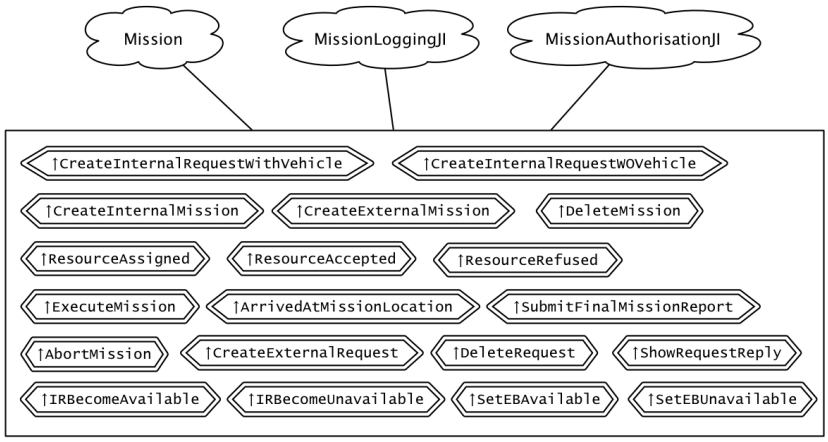


Figure 42.2: Behavioural diagram of package *MissionWithJoinInterface*.

Chapter 43

Package *CrisisWithAspects*

This chapter introduces package *CrisisWithAspects*, which adds required aspects to domain package *Crisis* (chapter 32). Essentially, it takes the *Crisis* package augmented with required join interfaces and composes required aspects. Package *CrisisWithAspects* extends packages *CrisisWithJI* (chapter 41), *LoggingCCCMS* (chapter 24), *AuthorisationCCCMS* (chapter 22) and *SessionMgmtCCCMS* (chapter 25).

43.1 Structure

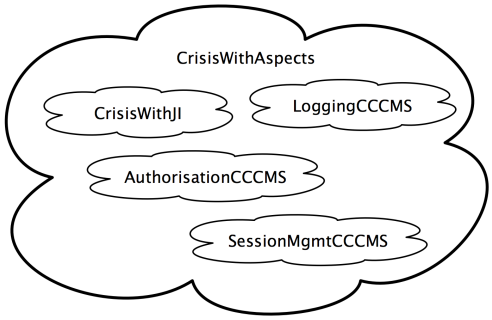


Figure 43.1: Package *CrisisWithAspects* extends packages *CrisisWithJi*, *LoggingCCCMS*, *AuthorisationCCCMS* and *SessionMgmtCCCMS*.

43.2 Behaviour

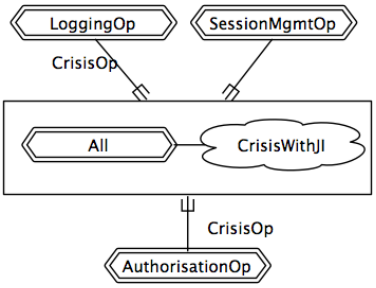


Figure 43.2: VCL behavioural diagram of package *CrisisWithAspects*. Global operations of this package are formed using *join extension*, where all operations of *CrisisWithJI* are extended with the behaviour captured in join contracts *LoggingOp*, *SessionMgmtOp* and *AuthorisationOp*.

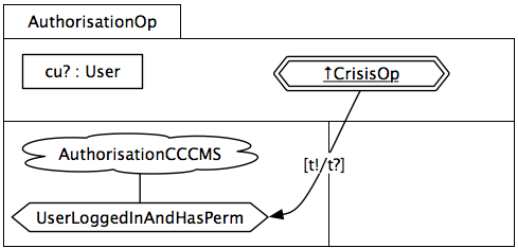


Figure 43.3: VCL contract diagram describing join contract *AuthorisationOp*. This extends one of the crisis operations being extended (represented by contract reference *CrisisOp*) with a precondition specified in package *AuthorisationCCCMS*, requiring that the user has required access permissions for current task (represented by input *t*? coming from *CrisisOp*).

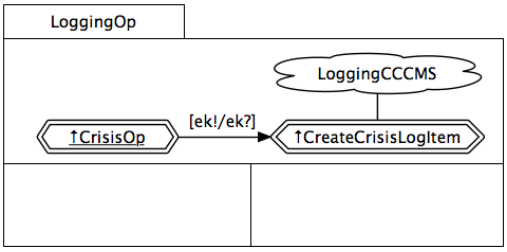


Figure 43.4: VCL contract diagram describing join contract *LoggingOp*. This extends one of the crisis operations being extended (represented by contract reference *CrisisOp*) with an operation of package *LoggingCCCMS* represented as contract reference *CreateCrisisLogItem*; this creates a log item of kind crisis for the current event (represented as input *ek*? coming from *CrisisOp*).

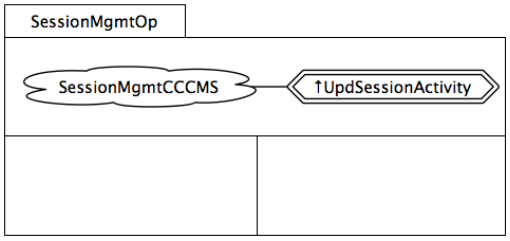


Figure 43.5: VCL contract diagram describing join contract *SessionMgmtOp*. This extends one of the crisis operations being extended with an operation of package *SessionMgmtCCCMS* represented as contract reference *UpdSessionActivity*; this updates the acitivity of current user's session.

Chapter 44

Package *MissionWithAspects*

This chapter introduces package *MissionWithAspects*, which adds required aspects to domain package *Mission* (chapter 32). Essentially, it takes the *Mission* package augmented with required join interfaces and composes required aspects. Package *MissionWithAspects* extends packages *MissionWithJI* (chapter 42), *LoggingCCCMS* (chapter 24), *AuthorisationCCCMS* (chapter 22) and *SessionMgmtCCCMS* (chapter 25).

44.1 Structure

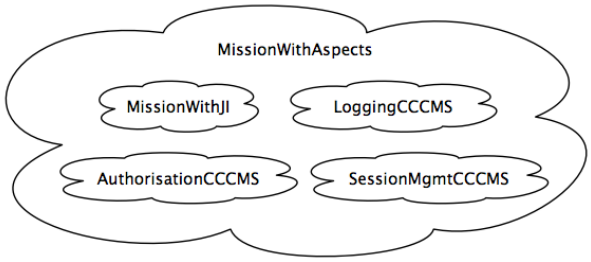


Figure 44.1: Package *MissionWithAspects* extends packages *MissionWithJi*, *LoggingCCCMS*, *AuthorisationCCCMS* and *SessionMgmtCCCMS*.

44.2 Behaviour

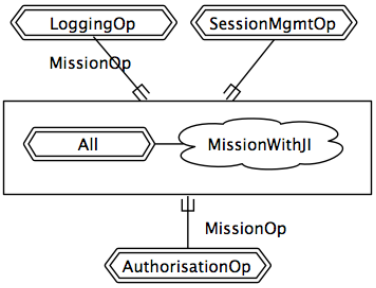


Figure 44.2: VCL behavioural diagram of package *MissionWithAspects*. Global operations of this package are formed using *join extension*, where all operations of *MissionWithJI* are extended with the behaviour captured in join contracts *LoggingOp*, *SessionMgmtOp* and *AuthorisationOp*.

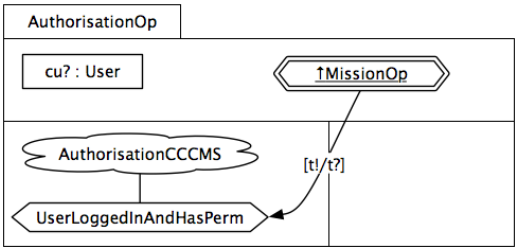


Figure 44.3: VCL contract diagram describing join contract *AuthorisationOp*. This extends one of the mission operations being extended (represented by contract reference *CrisisOp*) with a precondition specified in package *AuthorisationCCCMS*, requiring that the user has required access permissions for current task (represented by input *t*? coming from *MissionOp*).

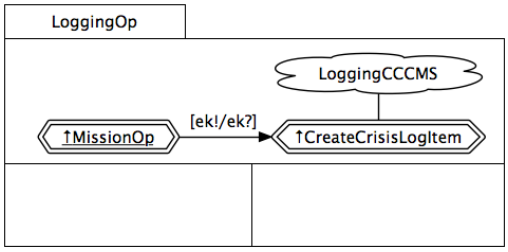


Figure 44.4: VCL contract diagram describing join contract *LoggingOp*. This extends one of the crisis operations being extended (represented by contract reference *MissionOp*) with an operation of package *LoggingCCCMS* represented as contract reference *CreateMissionLogItem*; this creates a log item of kind crisis for the current event (represented as input *ek*? coming from *MissionOp*).

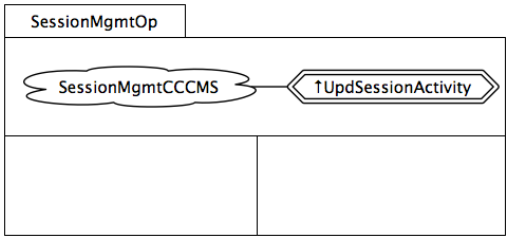


Figure 44.5: VCL contract diagram describing join contract *SessionMgmtOp*. This extends one of the mission operations being extended with an operation of package *SessionMgmtCCCMS* represented as contract reference *UpdSessionActivity*; this updates the acitivity of current user's session.

Part VII

Composing packages to define subsystems

Chapter 45

Package *CentralCCCMS*

This chapter introduces package *CentralCCCMS*, which includes all relevant packages that address the requirements of *CentralCCCMS* sub-system. Essentially, this package is made of those packages that augment domain packages with required aspects, together with some other packages representing generic concerns. Package *CentralCCCMS* extends packages *CrisisWithAspects* (chapter 43), *MissionWithAspects* (chapter 44), *AuthenticationOps* (chapter 7), *SecAuthorisationMgmt* (chapter 14), and *MappingDisplayCentralCCCMS*.

45.1 Structure

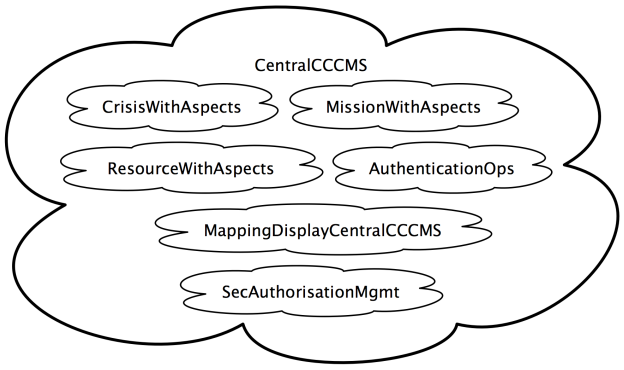


Figure 45.1: Package *CentralCCCMS* extends packages *CrisisWithAspects*, *MissionWithAspects*, *AuthenticationOps*, *SecAuthorisationMgmt*. and *MappingDisplayCentralCCCMS*.

45.2 Behaviour

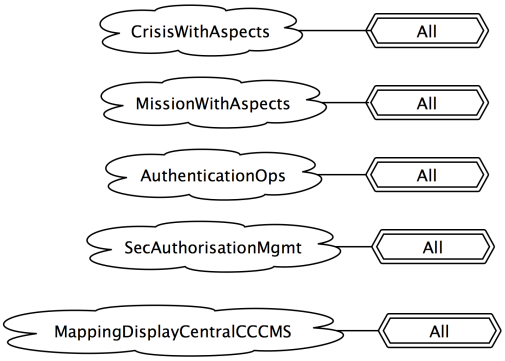


Figure 45.2: Behavioural diagram of package *System* where its global operations are defined by integral extension of global operations of packages *CrisisWithAspects*, *MissionWithAspects*, *AuthenticationOps* and *SecAuthorisationMgmt*.

Chapter 46

Package *MobCCCMS*

This chapter introduces package *MobCCCMS*, which includes all relevant packages that address the requirements of *MobCCCMS* sub-system. Package *MobCCCMS* extends packages *LocationTrackingMobCCCMS*, *MissionInfoDisplay*, *CrisisInfoDisplay*, *MobCCMSGUI*, and *MappingDisplayCentralCCCMS*.

46.1 Structure

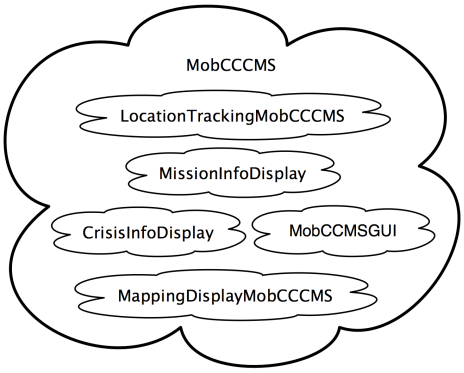


Figure 46.1: Package *MobCCCMS* extends packages *CrisisWithAspects*, *MissionWithAspects*, *AuthenticationOps*, *SecAuthorisationMgmt*. and *MappingDisplayCentralCCCMS*.

46.2 Behaviour

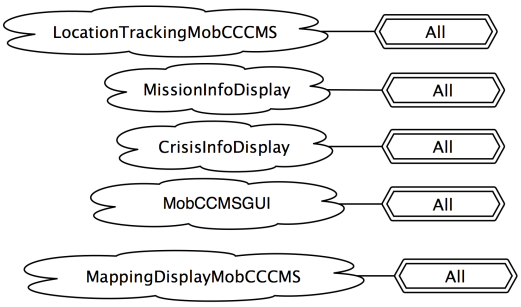


Figure 46.2: Behavioural diagram of package *System* where its global operations are defined by integral extension of global operations of packages *CrisisWithAspects*, *MissionWithAspects*, *AuthenticationOps* and *SecAuthorisationMgmt*.

References

[AK09] Nuno Amálio and Pierre Kelsen. The abstract syntax of structural VCL. Technical Report TR-LASSY-09-02, Laboratory of Advanced Software Systems, University of Luxembourg, 2009. available at `http://vcl.gforge.uni.lu/doc/abs-syn-strt-vcl-report.pdf`.

[AKM10] Nuno Amálio, Pierre Kelsen, and Qin Ma. The visual contract language: abstract modelling of software systems visually, formally and modularly. Technical Report TR-LASSY-10-01, Laboratory of Advanced Software Systems, University of Luxembourg, 2010. available at `http://vcl.gforge.uni.lu/doc/vcl-tech-rep.pdf`.

[KGM09] Jörg Kienzle, Nicolas Guelfi, and Sadaf Mustafiz. Crisis management systems a case study for aspect-oriented modeling. Technical Report SOCS-TR-2009.3, School of Computer Science. McGill University, 2009.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2), 1996.

Appendix A

Sample Z Specification of security packages

This chapter presents the Z that would be generated from the VCL diagrams that make the VCL security packages of the car-crash crisis management system (CCCMS).

A.1 Preamble

$$CLASS ::= UserCl \mid SessionCl \mid RoleCl \mid UserAdminCl$$

$$\begin{array}{|l} subCl : CLASS \leftrightarrow CLASS \\ abstractCl : \mathbb{P}\, CLASS \\ rootCl : \mathbb{P}\, CLASS \\ \hline subCl = \{(UserAdminCl, UserCl)\} \\ abstractCl = \{\} \\ rootCl = CLASS \setminus \operatorname{dom} subCl \end{array}$$

$$\begin{array}{|l} \mathbb{O} : CLASS \to \mathbb{P}_1\, OBJ \\ \mathbb{O}_x : CLASS \to \mathbb{P}_1\, OBJ \\ \hline \operatorname{disjoint} \mathbb{O}_x \\ \forall\, cl : CLASS \bullet \\ \quad \mathbb{O}\ cl = \mathbb{O}_x\ cl \cup \bigcup(\mathbb{O}_x(\!|\ (subCl^+)^{\sim}(\!|\ \{cl\}\ |\!)\ |\!)) \\ \forall\, cl, cl' : CLASS \mid cl \mapsto cl' \in subCl \bullet \\ \quad \mathbb{O}\ cl \subseteq \mathbb{O}\ cl' \end{array}$$

A.2 Package *Users*

A.2.1 Local Definitions of *User*

$$[UID, Name, Password]$$

$$\begin{array}{|l} maxPwMisses : \mathbb{N} \end{array}$$

$UserStatus ::= loggedIn \mid blocked \mid loggedOut$

UserDef

$uid : UID$
$name : Name$
$pw : Password$
$pwMisses : \mathbb{N}$
$status : UserStatus$

UserMaxPwMissesInv

$UserDef$

$pwMisses \leq maxPwMisses$

User

$UserDef$

$UserMaxPwMissesInv$

SUserDef

$sUser : \mathbb{P}(\mathbb{O}\ UserCl)$
$stUser : \mathbb{O}\ UserCl \pfun User$

$\mathrm{dom}\ stUser = sUser$

*Declu*1

$u1 : \mathbb{O}\ UserCl$

*Declu*2

$u2 : \mathbb{O}\ UserCl$

UsersEqual

$Declu1$
$Declu2$

$u1 = u2$

UsersHaveSameId

$SUserDef$
$Declu1$
$Declu2$

$(stUser\ u1).uid = (stUser\ u2).uid$

IDOfUsersUnique

$SUserDef$

$\forall\ Declu1;\ Declu2 \bullet UsersHaveSameId \Rightarrow UsersEqual$

SUser

$SUserDef$

$IDOfUsersUnique$

SUserGetUserGivenID

$SUser$
$u! : \mathbb{O}\ UserCl$
$uid? : UID$

$(stUser\ u!).uid = uid?$

A.2.2 Package Structural Definitions

Users

$$SUser$$

A.3 Package *UsersMgmt*

A.3.1 Local Definitions of *User*

UserNew

$$User'$$
$$uid? : UID$$
$$pw? : Password$$
$$name? : Name$$

$$uid' = uid?$$
$$name' = name?$$
$$pw' = pw?$$
$$pwMisses' = 0$$
$$status' = loggedOut$$

UserDelete

$$User$$

$$status \neq loggedIn$$

UserChangePassword

$$\Delta User$$
$$pwOld? : Password$$
$$pwNew? : Password$$

$$pw = pwOld? \land pw' = pwNew?$$
$$pw' = pw \land name' = name \land pwMisses' = pwMisses$$
$$status' = status$$

$\Phi SUserNew$

$$\Delta SUser$$
$$User'$$
$$u! : \mathbb{O}\ UserCl$$

$$u! \in \mathbb{O}_x\ UserCl \setminus sUser$$
$$sUser' = sUser \cup \{u!\}$$
$$stUser' = stUser \cup \{(u!, \theta User')\}$$

$\Phi SUserDel$

$$\Delta SUser$$
$$User$$
$$u? : \mathbb{O}\ UserCl$$

$$u? \in sUser$$
$$\theta User = stUser\ u?$$
$$sUser' = sUser \setminus \{u?\}$$
$$stUser' = \{u?\} \lhd stUser$$

$\Phi SUserUpd$

$\Delta SUser$
$\Delta User$
$u? : \mathbb{O}\ UserCl$

$u? \in sUser$
$\theta User = stUser\ u?$
$sUser' = sUser$
$stUser' = stUser \oplus \{(u?, \theta User')\}$

$SUserNew == \exists\ User' \bullet \Phi SUserNew \land UserNew$
$SUserDelete == \exists\ User \bullet \Phi SUserDel \land UserDelete$
$SUserChangePassword == \exists \Delta User \bullet \Phi SUserUpd \land UserChangePassword$

A.3.2 Package Structural Definitions

UsersMgmt

Users

A.3.3 Global Behaviour

$\Psi UsersMgmtCreateUser == \Delta UsersMgmt$
$UsersMgmtCreateUser == \Psi UsersMgmtCreateUser \land SUserNew \setminus (u!)$
$\Psi UsersMgmtRemoveUser == \Delta UsersMgmt$
$UsersMgmtRemoveUser == \Psi UsersMgmtRemoveUser \land SUserDelete$
$\quad \land SUserGetUserGivenID[u?/u!] \setminus (u?)$
$\Psi UsersMgmtChangeUserPassword == \Delta UsersMgmt$
$UsersMgmtChangeUserPassword == \Psi UsersMgmtChangeUserPassword$
$\quad \land SUserChangePassword \land SUserGetUserGivenID[u?/u!] \setminus (u?)$

A.4 Package *Authentication*

A.4.1 Local Definitions of *Session*

$[SID]$
$Time == \mathbb{Z}$

Session

$sid : SID$
$startTm : Time$
$lastTmActive : Time$

SSessionDef

$$sSession : \mathbb{P}(\mathbb{O}\ SessionCl)$$
$$stSession : \mathbb{O}\ SessionCl \pfun Session$$
$$\dom stSession = sSession$$

*Decls*1

$$s1 : \mathbb{O}\ SessionCl$$

*Decls*2

$$s2 : \mathbb{O}\ SessionCl$$

SessionsEqual

*Decls*1
*Decls*2
$$s1 = s2$$

SessionsHaveSameId

SSessionDef
*Decls*1
*Decls*2
$$(stSession\ s1).sid = (stSession\ s2).sid$$

IDOfSessionsUnique

SSessionDef
$$\forall\ Decls1;\ Decls2 \bullet SessionsHaveSameId \Rightarrow SessionsEqual$$

SSession

SSessionDef
IDOfSessionsUnique

A.4.2 Local Definitions of *HasSession*

HasSession

$$rHasSession : \mathbb{O}\ UserCl \rightrel{} \mathbb{O}\ SessionCl$$

A.4.3 Local Definitions of *User*

UserIsLoggedIn

SUser
$$cu? : \mathbb{O}\ UserCl$$
$$(stUser\ cu?).status = loggedIn$$

A.4.4 Package Structural Definitions

AuthenticationSt

$Users$
$SSession$
$HasSession$

Decls

$s : \mathbb{O}\ SessionCl$

Declu

$u : \mathbb{O}\ UserCl$

UserWithSession

$AuthenticationSt$
$Declu$
$Decls$

$(u, s) \in rHasSession$

UserHasSession

$AuthenticationSt$
$Declu$

$\exists\ Decls \bullet UserWithSession$

HasSessionIfLoggedIn

$AuthenticationSt$

$\forall\ Declu \bullet UserHasSession \Leftrightarrow UserIsLoggedIn[u/cu?]$

HasSessionGCnt

$AuthenticationSt$

$\text{mult}(rHasSession, sUser, sSession, ozo, \{\}, \{\})$

Authentication

$AuthenticationSt$

$HasSessionIfLoggedIn$
$HasSessionGCnt$

A.4.5 Global Behaviour

$AuthenticationUserIsLoggedIn == Authentication \land UserIsLoggedIn$

A.5 Package *AuthenticationOps*

A.5.1 Local Definitions of *User*

$LoginResult ::= loginOK \mid wrongPW \mid isBlocked$

UserLoginOk

$\Delta User$
$pw? : Password$
$r! : LoginResult$

$status = loggedOut$
$pw = pw?$
$pwMisses' = 0$
$status' = loggedIn$
$pw' = pw$
$name' = name$
$uid' = uid$
$r! = loginOK$

UserLogout

$\Delta User$

$status = loggedIn$
$status' = loggedOut$
$pwMisses' = pwMisses$
$pw' = pw$
$name' = name$
$uid' = uid$

UserLoginBlocked

$\Delta User$
$r! : LoginResult$

$status = blocked$
$status' = status$
$pwMisses' = pwMisses$
$pw' = pw \land name' = name$
$uid' = uid$
$r! = isBlocked$

UserLoginWrongPW

$\Delta User$
$pw? : Password$
$r! : LoginResult$

$status = loggedOut$
$pw \neq pw?$
$pwMisses < maxPwMisses$
$status' = status$
$pwMisses' = pwMisses + 1$
$pw' = pw \land name' = name$
$uid' = uid$
$r! = wrongPW$

UserLoginWrongPWToBlocked

$\Delta User$
$pw? : Password$
$r! : LoginResult$

$status = loggedOut$
$pw \neq pw?$
$pwMisses = maxPwMisses$
$status' = blocked$
$pwMisses' = pwMisses$
$pw' = pw \land name' = name \land uid' = uid$
$r! = wrongPW$

$UserLoginNotOk == UserLoginBlocked \lor UserLoginWrongPW \lor UserLoginWrongPWToBlocked$

$SUserLoginOk == \exists \Delta User \bullet \Phi SUserUpd \land UserLoginOk$

$SUserLogout == \exists \Delta User \bullet \Phi SUserUpd \land UserLogout$

$SUserLoginNotOk == \exists \Delta User \bullet \Phi SUserUpd \land UserLoginNotOk$

A.5.2 Local Definitions of *Session*

SessionNew

$$
\begin{array}{l}
Session' \\
sid? : SID \\
now? : Time \\
\hline
sid' = sid? \\
startTm' = now? \\
lastTmActive' = now?
\end{array}
$$

SessionDelete

$$
\begin{array}{l}
Session \\
\hline
\end{array}
$$

$\Phi SSessionNew$

$$
\begin{array}{l}
\Delta SSession \\
Session' \\
s! : \mathbb{O}\ SessionCl \\
\hline
s! \in \mathbb{O}_x\ SessionCl \setminus sSession \\
sSession' = sSession \cup \{s!\} \\
stSession' = stSession \cup \{(s! \mapsto \theta Session')\}
\end{array}
$$

$\Phi SSessionUpd$

$$
\begin{array}{l}
\Delta SSession \\
\Delta Session \\
s? : \mathbb{O}\ SessionCl \\
\hline
s? \in sSession \\
\theta Session = stSession\ s? \\
sSession' = sSession \\
stSession' = stSession \oplus \{(s?, \theta Session')\}
\end{array}
$$

$\Phi SSessionO$

$$
\begin{array}{l}
SSession \\
Session \\
s? : \mathbb{O}\ SessionCl \\
\hline
s? \in sSession \\
\theta Session = stSession\ s?
\end{array}
$$

$\Phi SSessionDel$

$\Delta SSession$
$Session$
$s? : \mathbb{O} SessionCl$

$s? \in sSession$
$\theta Session = stSession\ s?$
$sSession' = sSession \setminus \{s?\}$
$stSession' = \{s?\} \lhd stSession$

$SSessionNew == \exists\, Session' \bullet \Phi SSessionNew \land SessionNew$
$SSessionDelete == \exists\, Session \bullet \Phi SSessionDel \land SessionDelete$

A.5.3 Local Definitions of *HasSession*

HasSessionAddNew

$\Delta HasSession$
$u? : \mathbb{O}\ UserCl$
$s? : \mathbb{O}\ SessionCl$

$rHasSession' = rHasSession \cup \{(u?, s?)\}$

HasSessionDelGivenUser

$\Delta HasSession$
$u? : \mathbb{O}\ UserCl$

$rHasSession' = \{u?\} \lhd rHasSession$

HasSessionGetUserSession

$HasSession$
$u? : \mathbb{O}\ UserCl$
$s! : \mathbb{O}\ SessionCl$

$(u?, s!) \in rHasSession$

A.5.4 Package Structural Definitions

AuthenticationOps

$Authentication$

A.5.5 Global Behaviour

$\Psi AuthenticationOpsLoginOk == \Delta AuthenticationOps$

$AuthenticationOpsLoginOk == \Psi AuthenticationOpsLoginOk \land SUserLoginOk$
$\quad \land SSessionNew \land HasSessionAddNew[s!/s?] \setminus (sid?, s!)$

$\Psi AuthenticationOpsLoginNotOk ==$
$\quad \Delta AuthenticationOps \land \Xi SSession \land \Xi HasSession$

$AuthenticationOpsLoginNotOk ==$
$\quad \Psi AuthenticationOpsLoginNotOk \land SUserLoginNotOk$

$AuthenticationOpsLogin == SUserGetUserGivenID[u?/u!] \land$
$\quad (AuthenticationOpsLoginOk \lor AuthenticationOpsLoginNotOk) \setminus (u?)$

$\Psi AuthenticationOpsLogout == \Delta AuthenticationOps$

$AuthenticationOpsLogout == \Psi AuthenticationOpsLogout$
$\quad \land SUserGetUserGivenID[u?/u!]$
$\quad \land SUserLogout \land HasSessionDelGivenUser$
$\quad \land HasSessionGetUserSession[s?/s!]$
$\quad \land SessionDelete \setminus (u?, s?)$

A.6 Package *AuthenticationMgmt*

A.6.1 Local Definitions of *User*

UserReactivate

$\Delta User$

$status = blocked$
$status' = loggedOut$
$pwMisses' = 0$
$pw' = pw$
$name' = name \land uid' = uid$

UserBlock

$\Delta User$

$status = loggedOut$
$status' = blocked$
$pw' = pw \land pwMisses' = pwMisses$
$name' = name \land uid' = uid$

$SUserReactivate == \exists \Delta User \bullet \Phi SUserUpd \land UserReactivate$
$SUserBlock == \exists \Delta User \bullet \Phi SUserUpd \land UserBlock$

A.6.2 Package Structural Definitions

AuthenticationMgmt

$Authentication$

A.6.3 Global Behaviour

$$\Psi AuthenticationMgmtReactivateUser == \Delta AuthenticationMgmt \land \Xi HasSession \land \Xi SSession$$

$$AuthenticationMgmtReactivateUser == \Psi AuthenticationMgmtReactivateUser \land SUserGetUserGivenID[u?/u!] \land SUserReactivate \setminus (u?)$$

$$\Psi AuthenticationMgmtBlockUser == \Delta AuthenticationMgmt \land \Xi HasSession \land \Xi SSession$$

$$AuthenticationMgmtBlockUser == \Psi AuthenticationMgmtBlockUser \land SUserGetUserGivenID[u?/u!] \land SUserBlock \setminus (u?)$$

A.7 Package *SessionMgmt*

A.7.1 Local Definitions of *Session*

$$maxInactivityTm : Time$$

SessionUpdActiveTm

$$\Delta Session$$
$$now? : Time$$

$$lastTmActive' = now?$$
$$sid' = sid$$
$$startTm' = startTm$$

SessionIsExpired

$$Session$$
$$now? : Time$$

$$lastTmActive \geq now? - maxInactivityTm$$

$$SSessionUpdActiveTm == \exists \Delta Session \bullet \Phi SSessionUpd \land SessionUpdActiveTm$$

$$SSessionIsExpired == \exists Session \bullet \Phi SSessionO \land SessionIsExpired$$

A.7.2 Package Structural Definitions

SessionMgmt

$$AuthenticationOps$$

A.7.3 Global Behaviour

$$\Psi SessionMgmtUpdSessionActivity == \Delta SessionMgmt \land \Xi SUser \land \Xi HasSession$$

$$SessionMgmtUpdSessionActivityOk == \Psi SessionMgmtUpdSessionActivity \land HasSessionGetUserSession[cu?/u?] \land SUserReactivate \land \neg SSessionIsExpired \setminus (u?)$$

$$\Psi SessionMgmtUpdSessionActivityNotOk == \Delta SessionMgmt$$

$$ConnAuthenticationOpsLogout == [SUser;\ cu? : \mathbb{O}\ UserCl;\ uid? : UID \mid uid? = (stUser\ cu?).uid]$$

$$SessionMgmtUpdSessionActivityNotOk == \Psi SessionMgmtUpdSessionActivityNotOk \land ConnAuthenticationOpsLogout \land AuthenticationOpsLogout \land SSessionIsExpired \setminus (uid?)$$

$$SessionMgmtUpdSessionActivity == HasSessionGetUserSession[cu?/u?, s?/s!] \land (SessionMgmtUpdSessionActivityOk \lor SessionMgmtUpdSessionActivityNotOk) \setminus (s?)$$

A.8 Package *AccessControl*

A.8.1 Local Definitions of *Role*

$$[String]$$

Role

$$desc : String$$

SRoleDef

$$sRole : \mathbb{P}(\mathbb{O}\ RoleCl)$$
$$stRole : \mathbb{O}\ RoleCl \pfun Role$$
$$\dom stRole = sRole$$

*Declr*1

$$r1 : \mathbb{O}\ RoleCl$$

*Declr*2

$$r2 : \mathbb{O}\ RoleCl$$

RolesHaveSameDesc

$$SRoleDef$$
$$Declr1$$
$$Declr2$$
$$(stRole\ r1).desc = (stRole\ r2).desc$$

RolesEqual

$Declr1$
$Declr2$

$r1 = r2$

RoleDescUnique

SRoleDef

$\forall\, Declr1;\ Declr2 \bullet RolesHaveSameDesc \Rightarrow RolesEqual$

SRole

SRoleDef

RoleDescUnique

A.8.2 Local Definitions of *Task*

$[Task]$

A.8.3 Local Definitions of *HasRole*

HasRole

$rHasRole : \mathbb{O}\ UserCl \leftrightarrow \mathbb{O}\ RoleCl$

A.8.4 Local Definitions of *HasPerm*

HasPerm

$rHasPerm : \mathbb{O} RoleCl \leftrightarrow Task$

A.8.5 Package Structural Definitions

AccessControlSt

Users
SRole
HasRole
HasPerm

HasRoleGCnt

$AccessControlSt$

$\text{mult}(rHasRole, sUser, sRole, mm, \{\}, \{\})$

HasPermGCnt

$AccessControlSt$

$\text{mult}(rHasPerm, sRole, Task, mm, \{\}, \{\})$

AccessControl

$AccessControlSt$

$HasRoleGCnt$
$HasPermGCnt$

A.8.6 Global Behaviour

AccessControlUserHasPerm

$AccessControl$
$u? : \mathbb{O}\ UserCl$
$t? : Task$

$(u?, t?) \in rHasPerm$

A.9 Package *AccessControlMgmt*

A.9.1 Local Definitions of *Role*

RoleNew

$Role'$
$desc? : String$

$desc' = desc?$

RoleDelete

$Role$

$\Phi SRoleNew$

$\Delta SRole$
$Role'$
$r! : \mathbb{O}\ RoleCl$

$r! \in \mathbb{O}_x\ RoleCl \setminus sRole$
$sRole' = sRole \cup \{r!\}$
$stRole' = stRole \cup \{(r! \mapsto \theta Role')\}$

$\Phi SRoleDel$

$\Delta SRole$
$Role$
$r? : \mathbb{O}\ RoleCl$

$r? \in sRole$
$\theta Role = stRole\ r?$
$sRole' = sRole \setminus \{r?\}$
$stRole' = \{r?\} \lhd stRole$

$SRoleNew == \exists\, Role\ ' \bullet \Phi SRoleNew \land RoleNew$
$SRoleDelete == \exists\, Role \bullet \Phi SRoleDel \land RoleDelete$

A.9.2 Local Definitions of *HasRole*

HasRoleAddNew

$\Delta HasRole$
$u? : \mathbb{O}\ UserCl$
$r? : \mathbb{O} RoleCl$

$rHasRole' = rHasRole \cup \{(u?, r?)\}$

HasRoleDelete

$\Delta HasRole$
$u? : \mathbb{O}\ UserCl$
$r? : \mathbb{O}\ RoleCl$

$rHasRole' = rHasRole \setminus \{(u?, r?)\}$

HasRoleDelGivenRole

$\Delta HasRole$
$r? : \mathbb{O}\ RoleCl$

$rHasRole' = rHasRole \rhd \{r?\}$

A.9.3 Local Definitions of *HasPerm*

HasPermAddNew

$\Delta HasPerm$
$r? : \mathbb{O}\ RoleCl$
$t? : Task$

$rHasPerm' = rHasPerm \cup \{(r?, t?)\}$

HasPermDelete

$\Delta HasPerm$
$r? : \mathbb{O}\ RoleCl$
$t? : Task$

$rHasPerm' = rHasPerm \setminus \{(r?, t?)\}$

A.9.4 Package Structural Definitions

AccessControlMgmt

AccessControl

A.9.5 Global Behaviour

$\Psi AccessControlMgmtCreateRole ==$
$\Delta AccessControlMgmt \land \Xi Users \land \Xi HasRole \land \Xi HasPerm$

$AccessControlMgmtCreateRole == \Psi AccessControlMgmtCreateRole$
$\land\ SRoleNew \setminus (r!)$

$\Psi AccessControlMgmtDelRole == \Delta AccessControlMgmt \land \Xi Users \land \Xi HasPerm$

$AccessControlMgmtDelRole == \Psi AccessControlMgmtDelRole \land SRoleDelete$
$\land\ HasRoleDelGivenRole$

$\Psi AccessControlMgmtAssignUserRole ==$
$\Delta AccessControlMgmt \land \Xi Users \land \Xi Role \land \Xi HasPerm$

$AccessControlMgmtAssignUserRole == \Psi AccessControlMgmtAssignUserRole$
$\land\ HasRoleAddNew$

$\Psi AccessControlMgmtDelRoleAssignment ==$
$\Delta AccessControlMgmt \land \Xi Users \land \Xi Role \land \Xi HasPerm$

$AccessControlMgmtDelRoleAssignment ==$
$\Psi AccessControlMgmtDelRoleAssignment \land HasRoleDelete$

$\Psi AccessControlMgmtAddPermission ==$
$\Delta AccessControlMgmt \land \Xi Users \land \Xi Role \land \Xi HasRole$

$AccessControlMgmtAddPermission == \Psi AccessControlMgmtAddPermission$
$\land\ HasPermAddNew$

$\Psi AccessControlMgmtDelPermission ==$
$\Delta AccessControlMgmt \land \Xi Users \land \Xi Role \land \Xi HasRole$

$AccessControlMgmtDelPermission == \Psi AccessControlMgmtDelPermission$
$\land\ HasPermDelete$

A.10 Package *Authorisation*

A.10.1 Package Structural Definitions

Authorisation

Authentication
AccessControl

A.10.2 Global Behaviour

AuthorisationUserLoggedInAndHasPerm

Authorisation
$cu? : \mathbb{O}\ UserCl$
$t? : Task$

$AccessControlUserHasPerm[cu?/u?]$
$AuthenticationUserIsLoggedIn$

A.11 Package *SysAdmin*

A.11.1 Local Definitions of *UserAdmin*

UserAdmin

User

SUserAdmin

$sUserAdmin : \mathbb{P}(\mathbb{O}\ UserAdminCl)$
$stUserAdmin : \mathbb{O}\ UserAdminCl \pfun UserAdmin$

$\mathrm{dom}\ stUserAdmin = sUserAdmin$

SUserAdminIsUser

SUser
SUserAdmin

$sUserAdmin \subseteq sUser$
$\forall\, oUA : sUserAdmin \bullet$
$\quad (\lambda\, UserAdmin \bullet \theta User)(stUserAdmin\ oUA) = stUser\ oUA$

A.11.2 Package Structural Definitions

SysAdmin

$$Users$$
$$SUserAdmin$$

$$SUserAdminIsUser$$

A.11.3 Global Behaviour

SysAdminIsSysAdmin

$$SysAdmin$$
$$cu? : \mathbb{O}\ UserCl$$

$$cu? \in sUserAdmin$$

A.12 Package *SecAuthorisationMgmt*

A.12.1 Package Structural Definitions

SecAuthorisationMgmt

$$AccessControlMgmt$$
$$AuthenticationMgmt$$
$$UsersMgmt$$
$$SysAdmin$$

A.12.2 Global Behaviour

$$\Psi SecAuthorisationMgmtCreateUser == \Delta SecAuthorisationMgmt \land \Psi UsersMgmtCreateUser \land \Xi SSession \land \Xi SUserAdmin \land \Xi SRole \land \Xi HasRole \land \Xi HasSession \land \Xi HasPerm$$

SecAuthorisationMgmtCreateUser

$$\Psi SecAuthorisationMgmtCreateUser$$
$$UsersMgmtCreateUser$$
$$cu? : \mathbb{O}\ UserCl$$

$$SysAdminIsSysAdmin$$
$$AuthenticationUserIsLoggedIn$$

$$\Psi SecAuthorisationMgmtRemoveUser == \Delta SecAuthorisationMgmt \land \Xi SSession \land \Xi SUserAdmin \land \Xi SRole \land \Xi HasRole \land \Xi HasSession \land \Xi HasPerm$$

SecAuthorisationMgmtRemoveUser

$\Psi SecAuthorisationMgmtRemoveUser$
$UsersMgmtRemoveUser$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$$\Psi SecAuthorisationMgmtChangeUserPassword == \Delta SecAuthorisationMgmt \\ \land\ \Xi SSession \land \Xi SUserAdmin \land \Xi SRole \land \Xi HasRole \\ \land\ \Xi HasSession \land \Xi HasPerm$$

SecAuthorisationMgmtChangeUserPassword

$\Psi SecAuthorisationMgmtChangeUserPassword$
$UsersMgmtRemoveUser$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$$\Psi SecAuthorisationMgmtReactivateUser == \Delta SecAuthorisationMgmt \\ \land\ \Psi AuthenticationMgmtReactivateUser \land \Xi SUserAdmin \land \Xi SRole \\ \land\ \Xi HasRole \land \Xi HasPerm$$

SecAuthorisationMgmtChangeReactivateUser

$\Psi SecAuthorisationMgmtReactivateUser$
$AuthenticationMgmtReactivateUser$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$$\Psi SecAuthorisationMgmtBlockUser == \Delta SecAuthorisationMgmt \\ \land\ \Psi AuthenticationMgmtBlockUser \land \Xi SUserAdmin \land \Xi SRole \\ \land\ \Xi HasRole \land \Xi HasPerm$$

SecAuthorisationMgmtBlockUser

$\Psi SecAuthorisationMgmtBlockUser$
$AuthenticationMgmtBlockUser$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtCreateRole == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtCreateRole \land \Xi SUserAdmin$

SecAuthorisationMgmtCreateRole

$\Psi SecAuthorisationMgmtCreateRole$
$AccessControlMgmtCreateRole$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtDelRole == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtDelRole \land \Xi SUserAdmin$

SecAuthorisationMgmtDelRole

$\Psi SecAuthorisationMgmtDelRole$
$AccessControlMgmtDelRole$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtAssignUserRole == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtAssignUserRole \land \Xi SUserAdmin$

SecAuthorisationMgmtAssignUserRole

$\Psi SecAuthorisationMgmtAssignUserRole$
$AccessControlMgmtAssignUserRole$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtDelRoleAssignment == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtDelRoleAssignment \land \Xi SUserAdmin$

SecAuthorisationMgmtDelRoleAssignment

$\Psi SecAuthorisationMgmtDelRoleAssignment$
$AccessControlMgmtDelRoleAssignment$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtAddPermission == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtAddPermission \land \Xi SUserAdmin$

SecAuthorisationMgmtAddPermission

$\Psi SecAuthorisationMgmtAddPermission$
$AccessControlMgmtAddPermission$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

$\Psi SecAuthorisationMgmtDelPermissionn == \Delta SecAuthorisationMgmt$
$\quad \land \Psi AccessControlMgmtDelPermission \land \Xi SUserAdmin$

SecAuthorisationMgmtDelPermission

$\Psi SecAuthorisationMgmtDelPermissionn$
$AccessControlMgmtDelPermission$
$cu? : \mathbb{O}\ UserCl$

$SysAdminIsSysAdmin$
$AuthenticationUserIsLoggedIn$

A.13 Toolkit

$[OBJ]$

$MultTy ::= mm \mid mo \mid om \mid mzo \mid zom \mid oo \mid zozo \mid zoo \mid ozo \mid ms \mid sm \mid ss$
$\quad \mid so \mid os \mid szo \mid zos$

$[X, Y]$

$\text{mult}\_ : \mathbb{P}((X \leftrightarrow Y) \times \mathbb{P}\, X \times \mathbb{P}\, Y \times \mathit{MultTy} \times \mathbb{F}\, \mathbb{N} \times \mathbb{F}\, \mathbb{N})$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, mm, s_1, s_2)) \Leftrightarrow r \in sx \leftrightarrow sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, mo, s_1, s_2)) \Leftrightarrow r \in sx \to sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, om, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \to sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, mzo, s_1, s_2)) \Leftrightarrow r \in sx \pfun sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, zom, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \pfun sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, oo, s_1, s_2)) \Leftrightarrow r \in sx \rightarrowtail sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, zozo, s_1, s_2)) \Leftrightarrow r \in sx \pinj sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, zoo, s_1, s_2)) \Leftrightarrow r \in sx \rightarrowtail sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, ozo, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \rightarrowtail sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, ms, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, mm, s_1, s_2))$
$\land\ (\forall\, x : \text{dom}\, r \bullet \#(\{x\} \lhd r) \in s_1)$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, sm, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, mm, s_1, s_2))$
$\land\ (\forall\, y : \text{ran}\, r \bullet \#(r \rhd \{y\}) \in s_1)$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, ss, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, ms, s_1, \{\}))$
$\land\ (\text{mult}(r, sx, sy, sm, s_2, \{\}))$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, so, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, mo, s_1, s_2))$
$\land\ (\text{mult}(r, sx, sy, sm, s_1, s_2))$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, os, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, om, \{\}, \{\}))$
$\land\ (\text{mult}(r, sx, sy, ms, s_1, \{\}))$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, szo, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, mzo, \{\}, \{\}))$
$\land\ (\text{mult}(r, sx, sy, sm, s_1, \{\}))$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(\text{mult}(r, sx, sy, zos, s_1, s_2)) \Leftrightarrow (\text{mult}(r, sx, sy, zom, \{\}, \{\}))$
$\land\ (\text{mult}(r, sx, sy, ms, s_1, \{\}))$