Discovery of Stable Domain Abstractions for Reusable Pointcut Interfaces: Common Case Study for AO Modeling

Dimitri Van Landuyt,
Eddy Truyen,
Wouter Joosen
Report CW 560, Aug 2009
Discovery of Stable Domain Abstractions for Reusable Pointcut Interfaces: Common Case Study for AO Modeling

Dimitri Van Landuyt,
Eddy Truyen,
Wouter Joosen
Report CW560, Aug 2009

Department of Computer Science, K.U.Leuven

Abstract

This document presents the application of our method to discover stable domain abstractions for designing reusable pointcut interfaces in a car crash management system, a.k.a. the “common case study for aspect-oriented modeling”. It provides a detailed account of how we applied this method in this particular case study. More specifically, it presents and discusses the seven activities of the method.

The requirements engineering activities (1 to 4) start from use cases and domain models. These activities involve re-engineering the use case models in order to discover and define stable abstractions for the car crash management domain.

The software architecture activities (5 to 7) involve the design of pointcut interfaces in an aspect-oriented architecture and the composition of aspects to base modules through these pointcut interfaces. This results in an AO architecture with reusable pointcut interfaces, as they are based on stable domain abstractions.
Discovery of Stable Domain Abstractions for Reusable Pointcut Interfaces: Common Case Study for AO Modeling

Dimitri Van Landuyt, Eddy Truyen, Wouter Joosen

Abstract

This document presents the application of our method to discover stable domain abstractions for designing reusable pointcut interfaces in a car crash management system, a.k.a. the "common case study for aspect-oriented modeling". It provides a detailed account of how we applied this method in this particular case study. More specifically, it presents and discusses the seven activities of the method.

The requirements engineering activities (1 to 4) start from use cases and domain models. These activities involve re-engineering the use case models in order to discover and define stable abstractions for the car crash management domain.

The software architecture activities (5 to 7) involve the design of pointcut interfaces in an aspect-oriented architecture and the composition of aspects to base modules through these pointcut interfaces. This results in a AO architecture with reusable pointcut interfaces, as they are based on stable domain abstractions.

1 Index

Contents

1 Index 1

2 Introduction 6

2.1 Method to Discover Stable Domain Abstraction to Design Reusable Pointcut Interfaces . . 6
2.2 Common Case Study for AO Design: a Car Crash Management System CCMS . . . . . . . . 7

I Application of the method: iteration 1 8

3 Activity 1. Gather domain models and use cases 8

3.1 Selection of requirements .................................................. 8
3.1.1 Introduction ................................................................. 8
3.1.2 Domain Models ........................................................... 8
3.1.3 Use Cases ................................................................. 10
3.1.4 Other Requirements .................................................... 10
3.2 Additional Domain Analysis and Use Case Engineering iteration ................................ 11
3.2.1 Introduction ................................................................. 11
3.2.1.1 Main decisions, changes and interpretations. ................. 12
3.2.2 Domain Analysis ......................................................... 12
3.2.2.1 Car Crash Resolution ............................................. 12
3.2.2.2 Vehicles ............................................................... 13
3.2.2.3 Media ................................................................. 14
3.2.3 Persons ............................................................... 14
3.2.3.1 External Resource Providers .................................... 14
3.2.3.2 External Information Providers ................................. 15
3.2.3.3 Super Observer Mission ........................................... 15

1
3.2.3.4 First-Aid Mission .................................................. 16
3.2.3.5 Victim Transportation Mission .................................. 16
3.2.4 Use Cases ............................................................. 18
3.2.4.1 Use Case Diagram .................................................. 18
3.2.4.2 UC1: Indicate Availability ....................................... 19
3.2.4.3 UC2: Authenticate and Authorize .............................. 19
3.2.4.4 UC3: Capture Witness Report .................................. 20
3.2.4.5 UC4: Select Strategy ............................................. 21
3.2.4.6 UC5a: Request Observation ...................................... 22
3.2.4.7 UC5b: Request First-Aid ......................................... 22
3.2.4.8 UC5c: Request Victim Transportation ......................... 23
3.2.4.9 UC6a: Execute Super Observer Mission ...................... 23
3.2.4.10 UC6b: Execute First-Aid Mission ............................ 24
3.2.4.11 UC6c: Execute Victim Transportation Mission .......... 25
3.2.4.12 UC7: Update Weather Information ............................ 25
3.2.5 Usage Scenario ....................................................... 26

4 Activity 2. Identify and Separate Crosscutting Concerns ................. 27
4.1 A. Identification of Crosscutting Concerns .......................... 27
4.1.1 authentication check ................................................. 27
4.1.2 authorization check .................................................. 27
4.1.3 witness identity validation ......................................... 27
4.1.4 crisis confirmation ................................................... 28
4.1.5 strategy revision ...................................................... 28
4.1.6 real-time update ..................................................... 28
4.1.7 resource provider availability (update) .......................... 29
4.1.8 statistical logging .................................................... 29
4.1.9 Crosscutting concerns over use cases ............................ 30
4.2 B. Separation of crosscutting concerns ............................... 31
4.2.1 Base use cases ....................................................... 31
4.2.1.1 UC1: Change Availability State ............................... 31
4.2.1.2 UC2: Authenticate and Authorize ............................ 32
4.2.1.3 UC3: Capture Witness Report .................................. 32
4.2.1.4 UC4: Select Strategy ............................................. 33
4.2.1.5 UC5a: Request Observation ...................................... 34
4.2.1.6 UC5b: Request First-Aid ......................................... 34
4.2.1.7 UC5c: Request Victim Transportation ......................... 35
4.2.1.8 UC6a: Execute Super Observer Mission ...................... 35
4.2.1.9 UC6b: Execute First-Aid Mission ............................ 36
4.2.1.10 UC6c: Execute Victim Transportation Mission .......... 36
4.2.1.11 UC7: Update Weather Information ............................ 37
4.2.2 Extension use cases .................................................. 37
4.2.2.1 EXT_UC1: Check Authentication and Authorization ........ 37
4.2.2.2 EXT_UC2: Validate Witness Identity .......................... 38
4.2.2.3 EXT_UC3: Confirm Car Crash .................................. 38
4.2.2.4 EXT_UC4: Update Car Crash Information .................... 39
4.2.2.5 EXT_UC5: Revise Strategy ...................................... 39
4.2.2.6 EXT_UC6: Check Resource Provider Availability .......... 40
4.2.2.7 EXT_UC7: Set Resource Provider Status Unavailable .... 40
4.2.2.8 EXT_UC8: Set Resource Provider Status Available ....... 41
4.2.2.9 EXT_UC9: Log Action ........................................... 41
4.2.3 Use case diagram .................................................... 42
5 Activity 3. Generalize crosscutting extend-relationships

5.1 Use cases

5.1.1 Base use cases
5.1.1.1 UC1: Indicate Availability State
5.1.1.2 UC2: Authenticate and Authorize
5.1.1.3 UCa: Perform Crisis Management Action (abstract)
5.1.1.4 UCb: Perform Action Requiring CMS Employee Selection (abstract)
5.1.1.5 UC3: Capture Witness Report
5.1.1.6 UC4: Select Strategy
5.1.1.7 UC5: Request Resource (abstract)
5.1.1.8 UC5a: Request Observation
5.1.1.9 UC5b: Request First-Aid
5.1.1.10 UC5c: Request Victim Transportation
5.1.1.11 UCc: Send Car Crash Information Update (abstract)
5.1.1.12 UC6: Execute Mission (abstract)
5.1.1.13 UC6a: Execute Super Observer Mission
5.1.1.14 UC6b: Execute First-Aid Mission
5.1.1.15 UC6c: Execute Victim Transportation Mission
5.1.1.16 UC7: Update Weather Information

5.1.2 Extension use cases
5.1.2.1 EXT UC1: Check Authentication and Authorization
5.1.2.2 EXT UC2: Validate Witness Identity
5.1.2.3 EXT UC3: Confirm Car Crash
5.1.2.4 EXT UC4: Update Car Crash Information
5.1.2.5 EXT UC5: Revise Strategy
5.1.2.6 EXT UC6: Check Resource Provider Availability
5.1.2.7 EXT UC7: Set Resource Provider Status Unavailable
5.1.2.8 EXT UC8: Set Resource Provider Status Available
5.1.2.9 EXT UC9: Log Action

5.2 Use case diagram

6 Activity 4. Define Stable Domain Abstractions

6.1 Domain Abstraction Inventory
6.1.1 car crash information update received (short: update received)
6.1.2 authentication success
6.1.3 authentication failed
6.1.4 authorization success
6.1.5 authorization failed
6.1.6 witness identity known
6.1.7 witness identity validated
6.1.8 fake witness
6.1.9 car crash location known
6.1.10 car crash confirmed
6.1.11 fake car crash
6.1.12 strategy selected
6.1.13 employee selection
6.1.14 employee agreement
6.1.15 no employee available
6.1.16 car crash management action initiation (short: action initiation)
6.1.17 car crash management action performed (short: action performed)
6.1.18 car crash management action failed (short: action failed)

6.2 Updated domain models
7 Activities 5–7. Architecture Design for Reusable Pointcut Interfaces 60

7.1 Activity 5: Initial Architecture Design .................................................. 60
  7.1.1 Client-Server view ................................................................. 60
  7.1.2 Key Architectural Decisions .................................................... 60
  7.1.3 Components and Interfaces: client tier ....................................... 61
    7.1.3.1 Coordinator Workstation .................................................. 61
    7.1.3.2 External Information Provider ......................................... 61
    7.1.3.3 External Resource Provider .............................................. 61
  7.1.4 Components and Interfaces: client facade tier .............................. 62
    7.1.4.1 Coordinator Facade ......................................................... 62
    7.1.4.2 External Information Provider Facade .................................. 62
    7.1.4.3 External Resource Provider Facade ..................................... 63
  7.1.5 Components and Interfaces: business tier .................................... 63
    7.1.5.1 Authentication and Authorization ....................................... 63
    7.1.5.2 Authentication and Authorization Check ................................ 63
    7.1.5.3 Logger ........................................................................... 63
    7.1.5.4 Identity Validation .......................................................... 63
    7.1.5.5 Car Crash Confirmation ..................................................... 64
    7.1.5.6 CMS Employee Management ............................................... 64
    7.1.5.7 Car Crash Management ..................................................... 64
    7.1.5.8 Phone System ................................................................ 64
    7.1.5.9 Mission Execution ............................................................ 65
  7.1.6 Components and Interfaces: data tier .......................................... 65
    7.1.6.1 Car Crash DB ................................................................. 65
    7.1.6.2 CMS Employee DB .......................................................... 65
    7.1.6.3 Log DB ........................................................................ 65

7.2 Activity 6: Design Pointcut Interfaces ................................................. 66
  7.2.1 Design of Pointcut Interfaces ...................................................... 66
    7.2.1.1 update received ................................................................. 67
    7.2.1.2 authentication success ....................................................... 67
    7.2.1.3 authentication failure ......................................................... 67
    7.2.1.4 authorization success ......................................................... 67
    7.2.1.5 authentication failure ......................................................... 67
    7.2.1.6 witness identity known ....................................................... 67
    7.2.1.7 witness identity validated .................................................... 67
    7.2.1.8 fake witness ................................................................... 68
    7.2.1.9 car crash location known ................................................... 68
    7.2.1.10 car crash confirmed ......................................................... 68
    7.2.1.11 fake car crash ................................................................. 68
    7.2.1.12 strategy selected .............................................................. 68
    7.2.1.13 employee selection .......................................................... 68
    7.2.1.14 employee agreement ........................................................ 68
    7.2.1.15 no employee available ...................................................... 68
    7.2.1.16 action initiation ............................................................... 69
    7.2.1.17 action performed ............................................................. 69
    7.2.1.18 action failed .................................................................. 69

7.3 Activity 7: Design Aspectual Connectors .......................................... 69
  7.3.1 Design of Aspectual Connectors .................................................. 72
    7.3.1.1 Revise strategy ................................................................. 72
    7.3.1.2 Revise strategy ................................................................. 72
    7.3.1.3 Confirm car crash ............................................................. 72
    7.3.1.4 Update car crash ............................................................. 72
    7.3.1.5 Check Employee Availability ............................................. 72

II Application of the method: Iteration 2 .............................................. 72

8 Activity 1: additional requirement: criminal activity monitoring. .......... 73
9 Activities 2-4.

10 Activities 5-7.

III Discussion

11 Types of reuse.

11.0.1.6 Type-1: Reuse of existing pointcut interfaces.

11.0.1.7 Type-2: Reuse of aspect compositions.

11.0.1.8 Type-3: Reuse within pointcut interface hierarchies.

12 Time/Effort Spent


2 Introduction

This document presents the application of the method to discover stable domain abstractions for designing reusable pointcut interfaces [4] in the common case study for aspect-oriented design [2]. It serves to document how we applied this method in this concrete case study and provides all details of this.

Section 2.1 summarizes the method, and Section 2.2 shortly recaps the case study.

The remainder of this document is structured as follows: Section 3 discusses the requirements and domain analysis activity (activity 1). Then, Section 4 presents activity 2—the identification and separation of crosscutting concerns. Thereafter, Section 5 presents activity 3—generalization, and Section 6 the definition of Stable Domain Abstractions (activity 4). Finally, section 7 groups the final three architectural activities of our method (activities 5–7).

2.1 Method to Discover Stable Domain Abstraction to Design Reusable Pointcut Interfaces

This section briefly outlines the method. Figure 1 presents an activity diagram showing the sequence of steps to be performed. It depicts how the approach guides the software engineer through seven activities to discover stable domain abstractions and design a software architecture centered around reusable pointcut interfaces. Additionally, Table 1 presents the input and output artifacts per activity.

In activity 1, the requirements are gathered and modeled in a structured way, as use cases. Additionally, as our approach is domain-driven, a domain model for the application is defined in parallel.

In activity 2, the identification of crosscutting concerns takes place starting from the use case models. A variety of approaches have been proposed in literature that can be employed for this activity and we simply adopt those. After identifying the crosscutting concerns, the use cases are re-engineered so that each use case is affected by at most one concern. A suitable mechanism for this is the use case extension relationship, which connects an extension (aspect) use case to an extension point in the extended (base) use case.

These first two activities are mostly comparable to existing proposals and practices. Therefore, they are not to be considered part of the contribution of the paper.

In opposition, the subsequent two activities (3, 4) are essential and specific to our method. They comprise the key contribution of our work.
Table 1: Input and output artifacts per activity.

<table>
<thead>
<tr>
<th>Act.</th>
<th>Input</th>
<th>Artifacts</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>stakeholder requirements, domain analysis, technical/legal constraints, . . .</td>
<td></td>
<td>use cases, domain models</td>
</tr>
<tr>
<td>2</td>
<td>use cases; domain models</td>
<td></td>
<td>separated use cases; extension points</td>
</tr>
<tr>
<td>3</td>
<td>separated use cases; extension points; domain models</td>
<td></td>
<td>generalized use cases; abstract extension points</td>
</tr>
<tr>
<td>4</td>
<td>abstract extension points; domain models</td>
<td></td>
<td>domain abstractions</td>
</tr>
<tr>
<td>5</td>
<td>use cases; domain models; domain abstractions; existing component-based architecture (optional)</td>
<td></td>
<td>(constrained) component-based architecture</td>
</tr>
<tr>
<td>6</td>
<td>component-based architecture; domain abstractions</td>
<td></td>
<td>component-based architecture with pointcut interfaces</td>
</tr>
<tr>
<td>7</td>
<td>component-based architecture with pointcut interfaces</td>
<td></td>
<td>component-based architecture with pointcut interfaces and aspectual compositions</td>
</tr>
</tbody>
</table>

The concern separation effort of activity 2 typically results in highly coupled use case models. This complexity is inherently problem space complexity: it results from the crosscutting nature of an aspectual requirement. The goal of activity 3 is to generalize these use case models in order to manage this coupling more efficiently. This results in use case hierarchies of which the roots are abstract use cases with generalized extension points. These generalized extension points form the first indication for finding stable domain abstractions for crosscutting concern composition. Selecting a name for a generalized extension point involves paying ample attention to finding the domain abstraction that best describes the semantics of the extension point in question. The guiding principle is that the domain abstractions should be anchored in the domain model.

Activity 4 is the final requirements engineering activity in our approach. During this activity, the introduced abstract extension points lead to the definition of stable domain abstractions. Contextual information is added to the selected name to complete the definition of a stable domain abstraction. Contextual information refers to other domain concepts that are required for the particular composition-context. This information is obtained by analysing the data flow from the base use case to the aspect use case through the extension point, required to enable the aspect functionality.

The next three activities (5–6) leverage the discovered stable domain abstractions to design reusable pointcut interfaces. In this paper, we merely outline one possible implementation strategy. More concretely, we adopt a component-based architecture method and employ an aspect-oriented architecture notation (AO-ADL) to represent aspectual composition. Evidently, this particular implementation strategy is not mandatory.

In activities 5 to 7, the architecture is designed with the domain abstractions in mind. More specifically, the domain abstractions are mapped to pointcut interfaces. A pointcut interface has a scope that is limited to a module in the architecture (typically component, or a set thereof). It captures the artifacts in this module that match the specific domain abstraction; i.e. the named abstraction and the contextual information. Finally, aspectual compositions are specified in terms of the pointcut interfaces, leveraging upon state-of-the-art aspect-oriented architectural description languages (ADL’s).

Scope. As the method uses use cases to come up with suitable aspects and domain abstraction in these aspects, it is straightforward this method works only for functional concerns. Non-functional concerns or aspects –aspects that are hard or impossible to make explicit in use cases– are therefore out of scope of the method.

2.2 Common Case Study for AO Design: a Car Crash Management System CCMS

The “Common Case Study for AO Design”[2] is a realistic and thus very complex crisis management system. The goal of this system is to maximally automate the handling of several crises, one of which is the car crash. This system handles quite complex functionalities and is furthermore restricted by some
non-functional requirement with a high impact (such as real-time constraints, etc). In [2], several types of requirements and design artifacts are presented.

The goal of this case study is to serve as a common ground for applying AO Modeling techniques, and to enable comparing the results of these wide variety of AO modelling techniques and tools over the same case study.

In this document, we apply our method to discover stable domain abstractions for reusable pointcut interfaces. To this end, we focus mostly on the specific car crash management domain and functional requirements, and not the product-line view of a generic crisis management system). Also, we omit the presented design artifacts. The selection of requirements and scoping is further discussed in Section 3.1.

Part I

Application of the method: iteration 1

3 Activity 1. Gather domain models and use cases

In this activity, we performed two main actions:

1. First, we narrowed the scope of the application by selecting a suitable subset (Section 3.1).

2. Secondly, we performed an additional iteration of domain analysis and use case engineering (Section 3.1).

Throughout the document, we use red to annotate changes to the original requirements.

3.1 Selection of requirements

3.1.1 Introduction

The provided requirement documentation [2] was very broad in the sense that it contains a wide variety of artifacts: feature models, non-functional requirements, use cases, deployment diagrams, etc. Not all this input will serve for applying the method.

Therefore, as a first action, we selected a of subset of the provided requirements, based on:

- Suitability for the method. The method focuses only on functional crosscutting concerns. Therefore no non-functional requirements, meaning no requirements not expressable as some form of functionality. Also, no feature models.

- Scope. We selected the key requirements to speed up this iteration and reduce the effort to come to a result.

- Concreteness. There are some inconsistencies/deliberate gaps in the provided requirements. Instead of attempting to fill these gaps (and perhaps making incorrect assumptions/decisions) we omit these wherever possible. Also, we attempt to fill gaps or resolve inconsistencies and ambiguities where necessary.

Requirements not explicitly copied from [2] in this document are not selected to be part of the subset.

3.1.2 Domain Models

As starting point, we adopt the domain models as presented in [2]. These are copied in Figures 2 and 3. Concepts not selected for the subset are struckthrough with a red cross.
Figure 2: Domain Model 1 after selecting a subset

Figure 3: Domain Model 2 after selecting a subset
3.1.3 Use Cases

The use cases related to car crash resolution are selected. Figure 4 presents these use cases. Use cases not selected are struckthrough with a red cross. We will not repeat the detailed description of these use cases, please refer to [2] for them.

![Use Case diagram after selecting a subset](image)

For scoping reasons and to avoid incomplete requirements (see Section 3.1) we restricted ourselves to these three types of missions: a Super-Observer Mission, a First-Aid Mission and a Victim Transportation Mission, which is similar to the original Helicopter Transport Mission, but uses an Ambulance to transport a Victim to a Hospital. This is detailed further in Section 3.2.3.5.

3.1.4 Other Requirements

In addition to the domain models and use cases, we have selected other requirements from the original requirements document [2]. These are:

- “Updating the availability of CMSEmployees due to sickness or vacation”
- “Dependability-focussed requirement: Severe Weather Conditions: Bad weather makes helicopter transportation impossible.”

- **Real-time**
  - “The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.”

- **Security**
  - “The system shall define access policies for various classes of users. The access policy shall describe the components and information each class may add, access and update.”
  - “The system shall authenticate users on the basis of the access policies when they first access any components or information. If a user remains idle for 30 minutes or longer, the system shall require them to re-authenticate.”
• **Statistic Logging**
  
  “The system shall record the following statistical information on both on-going and resolved crises: rate of progression; average response time of rescue teams; individual response time of each rescue team; success rate of each rescue team; rate of casualties; success rate of missions.”

• **Safety**
  
  “The system shall monitor weather and terrain conditions at crisis site to ensure safe operation and withdrawal of rescue resources, and removal of civilians and casualties.”

• **Adaptability**
  
  “The system shall recommend alternate strategies for dealing with a crisis as the crisis conditions (e.g., weather conditions, terrain conditions, civilian or criminal activity) change.”

### 3.2 Additional Domain Analysis and Use Case Engineering Iteration

#### 3.2.1 Introduction

After the selection process, we found that:

1. The domain models were far from complete. Several important concepts in the problem domain (e.g. a Witness Report) were missing. However, these concepts were used throughout the text, e.g. in the use cases.

2. The use case space was decomposed in exactly one large use case (resolve crisis) which delegates certain activities to other use cases by means of use case inclusion: this can be considered to be quite a coarse-grained use case decomposition. This has a negative impact on use case cohesion and modularity: functionality not necessarily belonging together were grouped nonetheless (probably for reasons of space).

Therefore:

1. As the effectiveness of our method relies heavily on the quality and completeness of the domain models it acts upon, it was imperative to improved/completed the domain analysis in this iteration. This implied mostly that we had to explicitize many concepts that were already present in an implicit form.

2. Also, we improved (or at least changed) use case style. We strived for optimal separation of concerns in use cases: unrelated goals that were often presented as an alternative scenario in a use case are separated as much as possible. We state that the resulting set of use cases is at least equivalent to the use cases presented above (Section 3.1.3). Additionally, we explicitized the additional requirements selected earlier (see Section 3.1.4) and attempted to make them maximally visible in the use cases.

Some gaps were difficult to fill as several interpretations were possible. For completeness, we include these issues and discuss how they were resolved in the next paragraph. In most cases, we applied Occam’s Razor, i.e. selecting the most straightforward interpretation to fill the ambiguity in question.

Of course, throughout this activity, it was an ongoing concern not to actually stray from the original requirements specifications. Statements or decisions of which we are not fully certain (e.g. due to ambiguity in the original requirements documentation) are highlighted like ‘this’. We stayed very conservative in this regard: when in doubt, we highlighted.

Note that none of these activities are specific to our method: this is merely a pre-processing step to iron out ambiguities, gaps, and address vagueness. We claim that these are standard requirements engineering/domain analysis activities that would be necessary in the first place. In summary, the claim is that we are not changing the requirements in order to have our method to work.
3.2.1.1 Main decisions, changes and interpretations.

- We re-analysed the concept of a Resource and the distinction between Internal and External Resources and came to the following conclusion: the distinction between internal and external is much more useful when it comes to Resource Providers. A Resource is a something required for a Mission. For example, for a ‘Victim Transportation Mission’, the Resource is ‘Victim Transportation’. This resource can be provided by multiple (either internal, or external) Resource Providers. For example, an Ambulance can be used to transport a Victim, but also a Helicopter would be suited for that purpose.

- The concept of a Strategy is defined as follows: a Strategy is a set of specific Missions which –once executed– effectively address the Car Crash Crisis. The different actors of the System are tasked with keeping the System up-to-date in real-time. This allows the System to calculate a set of alternate strategies which have the same result (addressing the crisis) but differ in factors such as cost, effectiveness, risk, . . . . The System calculates the suitable strategies and presents them to the Coordinator. The Coordinator in turn selects the Strategy he finds most suitable in that situation. When the situation changes (e.g. weather changes, or traffic jams do not permit an ambulance to reach the crisis site in time), the System evaluates whether or not a change in Strategy would be appropriate.

- In terms of concrete Missions, we selected three of them: the Super Observer Mission, the Victim Transportation Mission and the First-Aid Mission. However, we simplified some of the details from the original use cases around concrete Missions [2].

- We included Weather information into the domain models and use cases, and describe how the Weather Information is an External Information Service that affects the System’s functionality. This is an explicitation of the rather informal “Adaptability”, “Safety” and “Dependability” requirements as selected in Section 3.1.4.

- We explicitly added “Availability Information” about a CMS Employee to the System: as the System’s functionality must vary depending on this information, this is visible in the use cases. Again, this is an explicitation of the rather informal requirement described in Section 3.1.4.

- We explicitized the requirements related to “Authorization” that were selected in Section 3.1.4. Similarly to “Authentication”, in the cases when the main actor is required to authenticate, the system also authorizes him; i.e. the system checks the user’s access rights before allowing or dis-allowing an action.

- We dropped the proposed generalization hierarchy in the use cases (around the use case ‘Execute Mission’ because of two reasons: (i) the presented generic use case ‘Execute Mission’ did not provide any commonalities (or a common frame for the child use cases), and it was only used to denote a commonality in goal, and (ii) the approach to discover stable domain abstractions for pointcut interfaces actively introduces a generalization hierarchy, therefore an existing generalization hierarchy might impede the method.

3.2.2 Domain Analysis

As discussed in Section 3.2.1, we performed another iteration of domain analysis with the goal of tightening the formulation, reducing vagueness and ambiguities and providing a more complete picture of the problem domain (in the scope of the selection done in Section 3.1). As the proposed case study pertains a very complex domain, we have split up the problem domain in a number of views, each of which presents a different facet of the problem domain. These views are consistent with each other.

3.2.2.1 Car Crash Resolution

This view presents the concepts related to car crash resolution, which is the main goal of the CCMS. The central driver in this process is the Coordinator. A Crisis (more specifically a Car Crash is identified when a Witness reports it to the Coordinator. The Witness Report contains all details related to the Car Crash.
The system then suggests one or more Strategies to address the Car Crash. A Strategy is a set of Missions that require several Resources, which are provided by Resource Providers (either internal or external).

The three concrete Mission types are illustrated in Sections 3.2.3.3 (a Super Observer Mission), 3.2.3.4 (a First-Aid Mission) and Section 3.2.3.5 (a Victim Transportation Mission).

**3.2.2.2 Vehicles**

This domain model details the types of vehicles that are in scope of the system. Therefore, this limits the types of car crashes that will be managed by this system. An additional constraint is that that a Car Crash involves at least one Car. For example, train crashes are out of scope. However, a train crashing into a car is in scope.
3.2.2.3 Media

We distinguish between three types of media that can be attached to a Witness Report: Picture, Sound, and Video. When a witness calls in a car crash, his call is recorded as a Sound clip.

![Figure 7: Domain Model: Media](image)

3.2.3 Persons

The different types of Persons or user roles that are relevant in the Car Crash Management System (CCMS) are depicted in this view. These include several Mobile Employee which in some case also assume the role of Internal Resource. All Workers have Credentials (a username/password combination) for authentication purposes and Access Rights for authorization purposes. The CMS Employee has an Availability State of which the System needs to be aware. This can either be Busy, Available or Away.

![Figure 8: Domain Model: Person](image)

3.2.3.1 External Resource Providers

The system will coordinate/cooperate with a number of External Resource Providers. Certain missions (such as transporting a victim to a hospital) are delegated to these third parties. In this case, the only External Resource Provider is the Helicopter Pilot.
3.2.3.2 External Information Providers

External Information Providers are additional sources of information. These include the traffic camera network (the Surveillance System), the Phone Company or the Weather Stations, providing information of the current weather and predictions of how the weather will evolve.

3.2.3.3 Super Observer Mission

A Super Observer is a Mobile Person that is sent to the crisis (car crash) location as soon as possible, observes and reports back the current progress and state of the crisis. This domain model illustrates this one concrete Mission type.
3.2.3.4 First-Aid Mission

A First-Aid Worker is a Mobile Person that is sent to the Crisis (Car Crash) to apply First-Aid techniques on the Victim(s). These could for example, rescuing the Victim from his Car.

3.2.3.5 Victim Transportation Mission

A Victim Transportation Mission is a Mission to transport a Victim from the Crisis site to the Hospital. Victim Transportation is an External Resource that is provided by a Helicopter Pilot External Resource. (Note that other External Resource Providers could be contacted to offer the same Resource, Victim Transportation. One such example is a Ambulance. The existence of such alternative that e.g. react differently to different conditions (weather, traffic jams, . . . ) illustrates the complexity of the decision-making process at the heart of this application.)
Figure 13: Domain Model: Victim Transportation Mission
3.2.4 Use Cases

3.2.4.1 Use Case Diagram

Figure 14: Use Case Diagram
### 3.2.4.2 UC1: Indicate Availability

- **ID:** UC1
- **Name:** Indicate Availability
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The CMS Employee indicates his availability to the system.
- **Preconditions:** The CMS Employee is authenticated.
- **Postconditions:** The System has stored the availability information of the CMS Employee and will take it into account when assigning the CMS Employee.

**Main Success Scenario:**

1. The CMS Employee indicates he wants to change his Availability Status.
2. The System presents a list of possible Availability Statuses.
3. The CMS Employee indicates his current Availability Status.
4. The System stores this information.

### 3.2.4.3 UC2: Authenticate and Authorize

- **ID:** UC2
- **Name:** Authenticate and Authorize
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The System wants to ensure identity of the CMS Employee and check his access rights.
- **Preconditions:** An CMS Employee has attempted to perform a certain action.
- **Postconditions:** The CMS Employee is authenticated and authorized to perform the requested action.

**Main Success Scenario:**

1. If the CMS Employee has not authenticated himself in the last 30 minutes,
   1. The System requests the CMS Employee to provide his credentials (e.g. uname/pwd)
   2. The CMS Employee provides his credentials
   3. The System validates the credentials
2. The System validates the access rights of the CMS Employee for the attempted action and grants the CMS Employee access.
   - The System logs this event.

**Extensions:**

1.3b The provided credentials are incorrect
   1. Repeat steps 1 and 2 three times.
   2. Use case fails.
      - The System logs this event.
2b The authenticated CMS Employee does not have the necessary access rights to perform the requested action.
   1. The use case fails, as the requested action is declined.
      - The System logs this event.
3.2.4.4 UC3: Capture Witness Report

- **ID:** UC3
- **Name:** Capture Witness Report
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Coordinator
- **Secondary Actor:** Phone Company, Surveillance System
- **Intention:** The Coordinator intends to create a crisis record based on the information obtained from the witness.
- **Postconditions:**
  - A car crash has been reported and is known to the system. An emergency level has been assigned.
  - The Witness Report is recorded.
  - The System has subscribed to Weather information for the Car Crash location.
- **Main Success Scenario:**
  1. Witness calls in to report a car crash
  2. The system searches for an available Coordinator and forwards the witness call to him
     - The System logs this event.
  3. The System checks whether the Coordinator is authenticated and authorized.
  4. The Coordinator answers the call
  5. The System sets the Coordinator’s Availability Status to “Unavailable”.
  6. The System starts recording the call
     - The System logs this event.
  7. Coordinator requests the Witness to provide his identification information. (Witness information includes the first name, last name, phone number, and address.)
  8. The Witness provides the requested identification information.
  9. Coordinator provides the identification information as reported by the witness to the System.
     - The System logs this event.
  10. In Parallel to the next steps:
    1 System contacts the Phone Company to verify witness information.
    2 Phone Company sends address/phone information to System.
    3 System validates information received from the Phone Company, and the Witness identity is confirmed.
        - The System logs this event.
  11. The Coordinator requests the witness to provide the location of the car crash
  12. The Witness provides the location information
  13. The Witness provides the System of location and type of crisis as reported by the witness.
      - The System logs this event.
  14. In parallel to the next steps:
    1 System requests video feed from Surveillance System.
    2 Surveillance System starts sending video feed to System.
    3 System starts displaying video feed for Coordinator.
    4 Coordinator confirms the crash visually and changes the state of the crisis to 'confirmed'
        - The System logs this event.
  15. System subscribes to Weather Information for the Car Crash Location by contacting the appropriate Weather Station.
  16. System provides Coordinator with a crisis-focused checklist to adhere to.
  17. Coordinator requests additional car crash information (e.g. number, types of Vehicles involved, vehicle identification)
  18. Witness provides the additional car crash information.
19. Coordinator provides crisis information to System as reported by the witness.
   - The System logs this event.
20. System assigns an initial emergency level to the crisis and sets the crisis status to active.
21. The system ends the call.
   - The System logs this event.
22. The system stops recording.

   - Extensions:

3b. Authentication and/or authorization check of the Coordinator failed.
   1 Include authenticate and authorize (UC2).
   1 Include authenticate and authorize (UC2).

   {3,4,5,6,7,8,9,10,11,12,13}b Call is disconnected.
   1 The system ignores the witness report (insufficient information)
   2 The system stops the recording

   {15,16,17,18}b The call is disconnected.
   1 The system attempts to re-dial the Witness.
   2 Witness answers the call.
   3 Systems continues the use case.

   {15,16,17,18}c The call is disconnected.
   1 The system attempts to re-dial the Witness.
   2 Witness does not answer the call.
   3 Systems continues the use case at step 17.

10.3b System validates information received from the Phone Company, and the Witness identity is
not confirmed.
   1 The system ignores the witness report (the witness is an impostor).
   2 The system stops the recording

14.4b Camera vision of the location is perfect, but Coordinator cannot confirm the situation that the
witness describes or the Coordinator determines that the witness is calling in a fake crisis.
   1 The system ignores the witness report (false information)
   2 The system stops the recording

3.2.4.5 UC4: Select Strategy

   - ID: UC4
   - Name: Select Strategy
   - Scope: Car Crash Crisis Management System
   - Primary Actor: Coordinator
   - Intention: The System manages the car crash by suggesting a number of alternative scenarios to
the Coordinator.

   - Preconditions:
     - A car crash was reported to the system.
     - The system has calculated a (number of) strategy/strategies to deal with this crisis.
     - Possibly, a strategy is already being followed which needs to be revised due to changing
       conditions (e.g. new car crash at same site, weather conditions).

   - Postconditions: A strategy has been selected and the system will follow it.

   - Main Success Scenario:
     1. The System searches for a Coordinator (based on expertise and availability)
     2. The system checks whether the Coordinator is authenticated and authorized
     3. The System sets the Coordinator's availability status to “Unavailable”.
4. The System presents the Coordinator with a number of alternative recommended Strategies based on the current information about the crisis and resources.
5. Coordinator selects one Strategy.
6. The System logs this event.
7. The System sets to Coordinator’s availability status to “Available”.

- Extensions:
  1b Authentication or authorization of the Coordinator failed: include Authenticate and Authorize (UC1).

3.2.4.6 UC5a: Request Observation

- **ID:** UC5a
- **Name:** Request Observation
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Super Observer
- **Intention:** The currently executing Super-Observation Mission requires some Observation as a Resource. Therefore, a Super-Observable that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a Super-Observable Mission requiring the requested Observation Resource.
- **Main Success Scenario:**
  1. The system selects an appropriate Super Observer based on the emergency level, availability, location. System sends the selected Super Observer a request for the required Observation Resource.
  2. The system checks whether the Super Observer is authenticated and authorized.
  3. Super Observer informs System that he will provide the requested Observation Resource and thus accepts the Super-Observation Mission.
     - The System logs this event.

- Extensions:
  1b No suitable Super Observer is available. Use case ends in failure.
     - The System logs this event.
  2b Authentication or authorization of the Super Observer failed: include Authenticate and Authorize (UC1).
  3b Super Observer is unavailable or unresponsive or Super Observer informs System that he cannot accept the mission.
     1. System selects the next appropriate Super Observer (Use case restarts at step 1).

3.2.4.7 UC5b: Request First-Aid

- **ID:** UC5b
- **Name:** Request First-Aid (Implements UC5)
- **Scope:** Car Crash Crisis Management
- **Primary Actor:** First-Aid Worker
- **Intention:** The currently executing First-Aid Mission requires some First-Aid as a Resource. Therefore, a First-Aid Worker that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a First-Aid Mission requiring the requested First-Aid Resource.
- **Main Success Scenario:**
  1. The system selects an appropriate First-Aid Worker based on the emergency level, availability, location. System sends the selected Super-Observable a request for the required First-Aid Resource.
2. The system checks whether the Resource Provider is authenticated and authorized.
3. First-Aid Worker informs System that he will provide the requested First-Aid Resource and thus accepts the First-Aid Mission.
   - The System logs this event.

- Extensions:
  1b No suitable First-Aid Worker is available. Use case ends in failure.
    - The System logs this event.
  2b Authentication or authorization of the First-Aid Worker failed: include Authenticate and Authorize (UC1).
  3b First-Aid Worker is unavailable or unresponsive or First-Aid Worker informs System that he cannot accept the mission.
    1 System selects the next appropriate First-Aid Worker (Use case restarts at step 1).

3.2.4.8 UC5c: Request Victim Transportation

- ID: UC5c
- Name: Request Victim Transportation
- Scope: Car Crash Crisis Management
- Primary Actor: Helicopter Pilot
- Intention: The currently executing Victim Transportation Mission requires some Victim Transportation as a Resource. Therefore, a Helicopter Pilot that has the necessary expertise is contacted.
- Preconditions: The System is following a certain Strategy which consists of a Victim Transportation Mission requiring the requested Victim Transportation Resource.
- Main Success Scenario:
  1. The system selects an appropriate Helicopter Pilot based on the emergency level, availability, location. System sends the selected Helicopter Pilot a request for the required Victim Transportation Resource.
  2. The system checks whether the Resource Provider is authenticated and authorized.
  3. Helicopter Pilot informs System that he will provide the requested Victim Transportation Resource and thus accepts the Victim Transportation Mission.
    - The System logs this event.
- Extensions:
  1b No suitable Helicopter Pilot is available. Use case ends in failure.
    - The System logs this event.
  2b Authentication or authorization of the Helicopter Pilot failed: include Authenticate and Authorize (UC1).
  3b Helicopter Pilot is unavailable or unresponsive or Helicopter Pilot informs System that he cannot accept the mission.
    1 System selects the next appropriate Helicopter Pilot (Use case restarts at step 1).

3.2.4.9 UC6a: Execute Super Observer Mission

- ID: UC6a
- Name: Execute Super Observer Mission
- Scope: Car Crash Crisis Management System
- Primary Actor: Super Observer
- Preconditions:
- The System has successfully requested the Observation Resource from a Super Observer
- Super Observer is at the Crisis Location.

- Main Success Scenario:
  1. The System checks whether the Super Observer is authenticated and authorized.
  2. The System requests the Super Observer to provide specific information about the Car Crash.
  3. The System sets Resource Provider's availability status to "Unavailable".
  4. The Super Observer provides the System with the requested Car Crash Information.
     - The System logs this event.
  5. The System adds this information the Car Crash information.
  6. The System sets to Super Observer’s availability status to “Available”.

- Extensions:
  1b Authentication or authorization of the Super Observer failed: include Authenticate and Authorize (UC1).
  5b Optionally, the System revises the current Strategy, Include: Select Strategy (UC4).

### 3.2.4.10 UC6b: Execute First-Aid Mission

- **ID:** UC6b
- **Name:** Execute First Aid Mission
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** First-Aid Worker
- **Intention:** The First-Aid Worker executes a First-Aid Mission by offering a the First-Aid Resource.

- **Preconditions:**
  - The System has successfully requested the First Aid Resource from a First Aid Worker
  - First Aid Worker is at the Crisis Location.

- **Main Success Scenario:**
  1. The System checks whether the First-Aid Worker is authenticated and authorized.
  2. The System briefs the First-Aid Worker with the concrete Mission details: the target Victim to rescue, its current Location, as well as the degree of urgency, potential medical history of the Victim, . . .
  3. The System sets to First-Aid Worker’s availability status to “Unavailable”.
  4. The First-Aid Worker rescues the Victim from the Car, offers First-Aid and reports this back to the System.
     - The System logs this event.
  5. The System adds this information the Car Crash information.
  6. The System sets to First-Aid Worker’s availability status to “Available”.

- **Extensions:**
  1b Authentication or authorization of the First-Aid Worker failed: include Authenticate and Authorize (UC1).
  5b Optionally, the System revises the current Strategy, Include: Select Strategy (UC4).
3.2.4.11 UC6c: Execute Victim Transportation Mission

- **ID**: UC6c
- **Name**: Execute Victim Transportation Mission
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: Helicopter Pilot
- **Intention**: The Helicopter Pilot executes a Victim Transportation Mission by offering a Victim Transportation Resource.

**Preconditions:**
- The System has successfully requested the Victim Transportation Resource from a Helicopter Pilot.

**Main Success Scenario:**
1. The System checks whether the Helicopter Pilot is authenticated and authorized.
2. The System briefs the Helicopter Pilot with the concrete Mission details: the target Victim to transport, its current Location, as well as the degree of urgency, and the destination Hospital of the Victim, ....
3. The System sets the Helicopter Pilot’s availability status to “Unavailable”.
4. The Helicopter Pilot flies to the Car Crash Location, picks up the Victim and transports it to the Hospital. Finally, she reports this back to the System.
   - The System logs this event.
5. The System adds this information to the Car Crash information.
6. The System sets the Helicopter Pilot’s availability status to “Available”.

**Extensions:**
1b Authentication or authorization of the Helicopter Pilot failed: include Authenticate and Authorize (UC1).
5b Optionally, the System revises the current Strategy, Include: Select Strategy (UC4).

3.2.4.12 UC7: Update Weather Information

- **ID**: UC9
- **Name**: Update Weather Information
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: Weather Station
- **Intention**: The Weather Station updates the CCMS in real-time about changing weather conditions.
- **Preconditions**: The System has indicated interest in Weather Information for a certain location.

**Main Success Scenario:**
1. The Weather Station pushes Weather Information updates to the System.
2. The system updates this information.

**Extensions:**
2b. Depending on the type of weather (or weather forecast) change, the System recommends a Strategy revision to the Coordinator (select strategy (UC4)).
3.2.5 Usage Scenario

Below, we present a short Usage Scenario, illustrating a concrete scenario in which the use cases (classifiers) presented in Section 3.2.4 are concretely instantiated.

On 10 June 2009, at 09:23, a blue Volvo drives into a parked bus at 200A, Celestijnenlaan, Heverlee (Belgium), right in front of the department of Computer Science. Driving speed was 65 km/h, and the driver of the Volvo was not wearing a seatbelt. Therefore, he endured a hard smack onto the steering wheel and is unconscious.

A student who was entering the building saw the accident happen and calls ‘100’ (the Belgian ‘911’). The call (at 09:24) is dispatched to the regional Car Crash Crisis Centre located in Leuven which runs this CCMS software. Coordinator John returned from his coffee break 9 minutes ago (at 09:13) and has indicated that he is available (use case Indicate Availability (UC1)). Therefore, the use case Capture Witness Report (UC3) is started: the Witness call starts recording, and John authenticates himself by holding his fingerprint against a fingerprint reader. John asks the student’s identification details, and enters these into the system. The System contacts the Phone Company to validate the phone number and identity information. This checks out. John asks the student to give the exact location of the where the car crash occurred. The system subscribes to the Weather Station for weather updates in this location. The Surveillance System is contacted for optional video feed or recent picture of that location. The Surveillance System is able to provide frames of 2 minutes ago and John the Coordinator can visually confirm the accident and the location. Furthermore, John asks the student to provide as much as additional information possible: are there multiple victims? What is the state of the victims, is there gas leaking from the car, etc. The student provides this information and John updates the status of the Witness Report.

The System calculates different Strategies for addressing this car crash. One of the scenarios requires the following Resources:

- ‘First-Aid’ to bring the Victim back to consciousness,
- “Victim Transportation” resource to transport the Victim to the Gasthuisberg Hospital nearby.
- the “Car Towing” resource is required to remove the damaged car.

The system proposes this particular strategy together with a few alternatives to John (use case Select Strategy). John selects the presented strategy.

The following three Missions are requested (UC5) and executed (UC6) in parallel:

- **First-Aid Mission**: The system contacts First-Aid Worker Henry, which has indicated his availability to the System earlier that day, and requests the “First-Aid” resource from him (Request First-Aid (UC5b)). Henry accepts the First-Aid Mission, and the executes the use case Execute First-Aid Mission (UC6b). Henry moves to the Car Crash Location, and finds the Victim. He reports the state of the Victim to the system. Then he brings the Victim back to consciousness and reports that this succeeded. Furthermore, he examines the Victim and identifies the severity of the injuries, and brings the System up-to-date. The Volvo driver has lost a lot of blood and requires urgent transport to the Hospital. This new information does not affect the current strategy, as urgent transportation was part of the original strategy.

- **Victim Transportation Mission**: The system contacts the Helicopter Pilot Jack, currently on a helipad on top of the Gasthuisberg hospital. The Helicopter pilot Jack is authenticated and accepts the Victim Transportation Mission (Request Victim Transportation (UC5c)). Thereafter, he takes off and moves to the location of the car crash with high urgency (Execute Victim Transportation Mission (UC6c)). He lands close to the car crash site where he picks up the Victim and transports him back to the Hospital. During this activity, he is in constant connection with the Coordinator John at the CCMS to keep him up-to-date on the progress.

- **Car Tow**, which is similar but out of scope.

During these different Missions, the System receives a Weather forecast for Heverlee indicating that light shower rain will fall in the next ten minutes (use case Update Weather Information (UC7)). The Coordinator responds by instructing the First-Aid worker to bring the Victim inside the Department of Computer Science building until the helicopter lands. In this case, the weather forecast does not require a change in strategy, as the weather change is not severe enough to keep the helicopter on the ground.

In summary, the victim was rescued from the vehicle, transported to the hospital by Helicopter Pilot and his damaged car was towed to the garage: the CCMS System has addressed this car crash in a timely manner.
4 **Activity 2. Identify and Separate Crosscutting Concerns**

4.1 **A. Identification of Crosscutting Concerns**

This Section introduces the different crosscutting concerns identified in the use cases of Section 3.2.4. For each crosscutting concern, we analyse its impact on the system. Also, we classify these crosscutting concerns in the aspect classification of Katz et al [3]. Also, for each of the identified concerns, we discuss how we identified it.

4.1.1 **authentication check**

One crosscutting concern is the *authentication check*. This involves checking whether or not the main actor of the use case is known to the system. If so, he is allowed to perform certain actions. If not, he is required to authenticate himself.

**Aspect classification.** This is a *regulative* aspect: it regulates the main use case flow, and only interferes this main flow when the authentication check fails.

**How identified.** This concern was identified by searching for recurring steps throughout the original use cases. In the use cases of Section 3.2.4, many use cases contain this pattern (in slight variation):

In the **Main success scenario**:

- The System checks whether the *main actor* is authenticated and authorized.

In the **Alternate flow**:

- Authentication and/or authorization check of the *main actor* failed.
  1. Include *authenticate and authorize (UC2)*.

Italics are used to denote the variabilities in these steps, wordgroups that are different per occurrence of this pattern. This consistent repetition of steps indicates a badly modularized concern in the use cases.

4.1.2 **authorization check**

Similar to the *authentication check*, before allowing a certain action to be performed by an authenticated actor, the system must also *authorize* the actor. This includes checking the actors’ access rights for the action or the resources involved in that action.

**Aspect classification.** Again, this is a *regulative* aspect: it regulates the main use case flow, by checking the access rights, and only interferes this main flow if authorization failed.

**How identified.** The *authorization check* concern is part of the pattern identified in the previous paragraph. In the original requirements of [2], the authorization concern was described only in the informal requirement “The system shall define access policies for various classes of users. The access policy shall describe the components and information each class may add, access and update.” Playing the advocates’ devil, one could argue that we deliberately added this to the use cases in a crosscutting fashion. However, our counterargument is that the way in which we represent this requirement in the use cases of Section 3.2.4, is much more explicit, and even more relevant. Based on the rather vague description quoted above, a software architect would not be constrained as much in terms of when authorization should be performed.

4.1.3 **witness identity validation**

A rather important functional concern in the Car Crash Management System (CCMS) is the *witness identity validation*: the system must ensure that the witness reporting a car crash is not pulling a prank, or providing false information. Therefore, the system must explicitly perform additional checks to validate the identity of the person on the phone. Moreover, this must be done in a timely manner: if this takes too long, it will impede the correct and timely handling of the car crash itself.
Aspect classification. This is a regulative aspect: it performs some background check (in parallel) and only interferes with the base flow when the identity validation fails.

How identified. This concern was identified by the realization that the Capture Witness Report use case is rather incohesive: some of the interactions presented in the use case do not serve the main goal of capturing witness reports, but are specific to a different concern, in this case witness identity validation. The description will benefit in modularity and use case cohesion if this concern is separated.

4.1.4 crisis confirmation

Very similar to the witness identity validation concern, crisis confirmation involves ensuring that the information provided by the witness is actually correct. Even if the identity of the witness is confirmed, this does not ensure correctness of the information provided by this witness. This concern involves actively checking the information provided by the witness about the car crash, currently by means of consulting traffic cameras, but in the foreseeable future this could include consulting satellite images, or calling other witnesses in the vicinity. This concern is essential as it increases the confidence the system can put in the witness report.

Aspect classification. Again, this is a regulative aspect: some background checks are performed in parallel, and the concern only interferes when crisis confirmation fails.

How identified. Similar to how the witness identity validation concern was identified.

4.1.5 strategy revision

Strategy revision is a functional concern that has a high impact on the use cases presented in Section 3.2.4. As stated in the original requirements (Adaptability): “The system shall recommend alternate strategies for dealing with a crisis as the crisis conditions”. To remain effective in a constantly changing environment, the system will constantly re-assess the current strategy. This occurs whenever the car crash situation changes (e.g. weather change, car crash situation change (e.g. another car crashes into crashed vehicles), unexpected traffic jams, ...). Assessing strategies takes into account many different factors, such as the safety risk.

Aspect classification. This is an invasive aspect: in some cases, the strategy revision takes over the main flow of execution; and alters the functioning of the system entirely. In the majority of the cases, however, it does not interrupt the main flow, but acts in parallel.

How identified. Again, a recurring pattern was identified in the use cases of Section 3.2.4:

In the Alternate flow:

x.b. Optionally, the System revises the current Strategy, Include: Select Strategy (UC4).

4.1.6 real-time update

The system must at all time have a recent view on the car crash situation and keep close eye on how it evolves. Therefore, many of the actors are required to update information in a fine-grained manner: the evolution of the car crash must be communicated to the CCMS in real-time as often as possible. This functional (but also slightly non-functional) concern has a large impact on the effectiveness of the CCMS system; when acting upon outdated information, it becomes very unlikely that the system will respond to the current state of the car crash in a suitable manner.

Aspect classification. This is an spectative aspect: it invasively impacts the functionality of the system, and the correct working of the system depends highly on this aspect.
How identified. Several use cases of Section 3.2.4 contain specific steps to updating the system. This pattern is as follows:

In the Main flow:

1. The main actor sends the system an update about the car crash.
2. The system updates this information.

Italics are used to denote the variabilities in these steps, wordgroups that are different per occurrence of this pattern. Also, this concern was part of the informal requirements in [2].

4.1.7 resource provider availability (update)

In order to respond swiftly to time-critical crises such as car crashes, the system must keep close track of its resource providers. By doing so, it can assess in a limited time frame and with little overhead in terms of communication which resource providers are available and which resource providers are not (and thus should not be contacted). To this end, the CCMS system requires these resource providers (and other actors) to actively keep their availability information up to date — and this at all time. This availability mechanism has a significant benefit over actively polling the availability of the resource providers when a car crash has occurred, as the decision time will be shortened. The functionality aspect of this concern is as follows: when the resource provider uses the system to perform a certain action (e.g. starting a Mission), the system automatically sets his availability state to ‘Unavailable’. When the action is finished, it is set back to ‘Available’.

Aspect classification. This is a spectative aspect; it interferes with the flow of actions in the main use case, but does not change it invasively.

How identified. Again, a recurring pattern was discovered:

In the Main flow:

1. The System sets to First-Aid Worker’s availability status to “Unavailable”.
2. Do something.
3. The System sets to First Aid Worker’s availability status to “Available”.

4.1.8 statistical logging

To analyse the overall performance, efficiency and effectiveness of the CCMS, a minute-to-minute log must be kept to allow auditing and post-fact analysis of what went wrong in case of system failure. More specifically, all interaction between actors and the system must be logged in detail, as well as the strategy proposed by the system and the progress of that strategy.

Aspect classification. This is a spectative aspect: it only observes the system, but does not interfere the functionality of the main flow.

How identified. This concern was identified by spotting recurring steps throughout the original use cases:

In the Main flow:

1. Do something.
   - The System logs this event.
4.1.9 Crosscutting concerns over use cases

Figure 15: Use Case Diagram - Concerns identified
Table 2: A use case–concern matrix that indicates for each use case whether or not it is affected by one of the concerns.

Table 2 and Figure 15 both present the mapping of the use cases to these concerns. They indicates for each use case by which concerns it is affected. From this table, it is straightforward to spot tangling and scattering of these concerns over the use cases. For example, the witness validation concern is not scattered over multiple use cases, but it is tangled in one use cases with other concerns (e.g., crisis confirmation). The resource provider availability concern is scattered over the use cases and tangled with other concerns in these use cases.

4.2 B. Separation of crosscutting concerns

The second part of Activity 2 is the separation of the crosscutting concerns identified in the previous Section. This results in a use case diagram in which the crosscutting nature of these concerns has manifested itself in terms of a large number of use case extension relationship (high coupling). To guide this separation, an algorithmic procedure was followed [].

4.2.1 Base use cases

4.2.1.1 UC1: Change Availability State

- **ID:** UC1
- **Name:** Indicate Availability State
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The CMS Employee indicates his Availability State to the system.
- **Preconditions:** The CMS Employee is authenticated.
- **Postconditions:** The System has stored the Availability State of the CMS Employee and will take it into account when assigning the CMS Employee.
- **Main Success Scenario:**
  1. The CMS Employee indicates he wants to change his Availability State.
  2. The System presents a list of possible Availability States.
  3. The CMS Employee indicates his current Availability State.
  4. The System changes the Availability State of the CMS Employee.
4.2.1.2 UC2: Authenticate and Authorize

- **ID:** UC2
- **Name:** Authenticate and Authorize
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The System wants to ensure identity of the CMS Employee and check his access rights.
- **Preconditions:** An CMS Employee has attempted to perform a certain action.
- **Postconditions:** The CMS Employee is authenticated and authorized to perform the requested action.

**Main Success Scenario:**

1. If the CMS Employee has not authenticated himself in the last 30 minutes,
   1. the System requests the CMS Employee to provide his credentials (e.g. uname/pwd)
   2. The CMS Employee provides his credentials
   3. **Extension point:** authentication success. The System validates the credentials
2. **Extension point:** authorization success. The System validates the access rights of the CMS Employee for the attempted action and grants the CMS Employee access.

**Extensions:**

1.3b The provided credentials are incorrect
   1. Repeat steps 1 and 2 three times.
   2. **Extension point:** authentication failure. Use case fails.
2b The authenticated CMS Employee does not have the necessary access rights to perform the requested action.
   1. **Extension point:** authorization failure. The use case fails, as the requested action is declined.

4.2.1.3 UC3: Capture Witness Report

- **ID:** UC3
- **Name:** Capture Witness Report
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Coordinator
- **Intention:** The Coordinator intends to create a crisis record based on the information obtained from the witness.

**Postconditions:**

- A car crash has been reported and is known to the system. An emergency level has been assigned.
- The Witness Report is recorded.
- The System has subscribed to Weather information for the Car Crash location.

**Main Success Scenario:**

1. Witness calls in to report a car crash
2. **Extension point:** witness call routed. The system forwards the witness call to a Coordinator
3. The Coordinator answers the call
4. **Extension point:** witness call initiated. The System starts recording the call
5. Coordinator requests the Witness to provide his identification information. (Witness information includes the first name, last name, phone number, and address.)
6. The Witness provides the requested identification information.
7. **Extension point:** witness identity known. Coordinator provides the identification information as reported by the witness to the System.
8. The Coordinator requests the witness to provide the location of the car crash
9. The Witness provides the location information
10. **Extension point: car crash location known.** The Witness provides the System of location
    and type of crisis as reported by the witness.
11. System subscribes to Weather Information for the Car Crash Location by contacting the
    appropriate Weather Station.
12. System provides Coordinator with a crisis-focused checklist to adhere to.
13. Coordinator requests additional car crash information (e.g. number, types of Vehicles involved,
    vehicle identification)
14. **Extension point: witness report complete.** Witness provides the additional car crash infor-
    mation.
15. Coordinator provides crisis information to System as reported by the witness.
16. System assigns an initial emergency level to the crisis and sets the crisis status to active.
17. **Extension point: witness call ended.** The system ends the call.
18. The system stops recording.

- **Extensions:**
  \{3,4,5,6,7,8,9,10\}
  1. Call is disconnected.
     1. The system ignores the witness report (insufficient information)
     2. The system stops the recording
  \{11,12,13\}
  1. The call is disconnected.
     1. The system attempts to re-dial the Witness.
     2. Witness answers the call.
     3. Systems continues the use case.
  \{11,12,13\}
  1. The call is disconnected.
     1. The system attempts to re-dial the Witness.
     2. Witness does not answer the call.
     3. Systems continues the use case at step 17.

### 4.2.1.4 UC4: Select Strategy

- **ID:** UC4
- **Name:** Select Strategy
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Coordinator
- **Intention:** The System manages the car crash by suggesting a number of alternative scenarios to
  the Coordinator.
- **Preconditions:**
  - A car crash was reported to the system.
  - The system has calculated a (number of) strategy/strategies to deal with this crisis.
- **Postconditions:** A strategy has been selected and the system will follow it.
- **Main Success Scenario:**
  1. **Extension point: coordinator selection.** The System searches for a Coordinator (based on
     expertise and availability)
  2. The System presents the Coordinator with a number of alternative recommended Strategies
     based on the current information about the crisis and resources.
  3. **Extension point: strategy selection.** Coordinator selects one Strategy.
4.2.1.5 UC5a: Request Observation

- **ID:** UC5a
- **Name:** Request Observation
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Super Observer
- **Intention:** The currently executing Super-Observation Mission requires some Observation as a Resource. Therefore, a Super-Observer that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a Super-Observer Mission requiring the requested Observation Resource.

**Main Success Scenario:**

1. **Extension point: super-observer selection.** The system selects an appropriate Super Observer based on the emergency level, availability, location. System sends the selected Super Observer a request for the required Observation Resource.
2. **Extension point: super-observer agreement.** Super Observer informs System that he will provide the requested Observation Resource and thus accepts the Super-Observation Mission.

**Extensions:**

1b **Extension point: insufficient super-observers.** No suitable Super Observer is available. Use case ends in failure.
2b Super Observer is unavailable or unresponsive or Super Observer informs System that he cannot accept the mission.
   1 System selects the next appropriate Super Observer (Use case restarts at step 1).

4.2.1.6 UC5b: Request First-Aid

- **ID:** UC5b
- **Name:** Request First-Aid *(Implements UC5)*
- **Scope:** Car Crash Crisis Management
- **Primary Actor:** First-Aid Worker
- **Intention:** The currently executing First-Aid Mission requires some First-Aid as a Resource. Therefore, a First-Aid Worker that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a First-Aid Mission requiring the requested First-Aid Resource.

**Main Success Scenario:**

1. **Extension point: first-aid worker selection.** The system selects an appropriate First-Aid Worker based on the emergency level, availability, location. System sends the selected First-Aid Observer a request for the required First-Aid Resource.
2. **Extension point: first-aid worker agreement.** First-Aid Worker informs System that he will provide the requested First-Aid Resource and thus accepts the First-Aid Mission.

**Extensions:**

1b **Extension point: insufficient first-aid workers.** No suitable First-Aid Worker is available. Use case ends in failure.
2b First-Aid Worker is unavailable or unresponsive or First-Aid Worker informs System that he cannot accept the mission.
   1 System selects the next appropriate First-Aid Worker (Use case restarts at step 1).
4.2.1.7 UC5c: Request Victim Transportation

- **ID:** UC5c
- **Name:** Request Victim Transportation
- **Scope:** Car Crash Crisis Management
- **Primary Actor:** Helicopter Pilot
- **Intention:** The currently executing Victim Transportation Mission requires some Victim Transportation as a Resource. Therefore, a Helicopter Pilot that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a Victim Transportation Mission requiring the requested Victim Transportation Resource.

**Main Success Scenario:**

1. **Extension point: helicopter pilot selection.** The system selects an appropriate Helicopter Pilot based on the emergency level, availability, location. System sends the selected Helicopter Pilot a request for the required Victim Transportation Resource.
2. **Extension point: helicopter pilot agreement.** Helicopter Pilot informs System that he will provide the requested Victim Transportation Resource and thus accepts the Victim Transportation Mission.

**Extensions:**

1b **Extension point: insufficient helicopters.** No suitable Helicopter Pilot is available. Use case ends in failure.
2b Helicopter Pilot is unavailable or unresponsive or Helicopter Pilot informs System that he cannot accept the mission.
   1 System selects the next appropriate Helicopter Pilot (Use case restarts at step 1).

4.2.1.8 UC6a: Execute Super Observer Mission

- **ID:** UC6a
- **Name:** Execute Super Observer Mission
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Super Observer
- **Intention:** The Super Observer executes a Super Observer Mission by offering a the Observation Resource.
- **Preconditions:**
  - The System has successfully requested the Observation Resource from a Super Observer
  - Super Observer is at the Crisis Location.

**Main Success Scenario:**

1. **Extension point: super-observer mission briefing.** The System requests the Super Observer to provide specific information about the Car Crash.
2. **Extension point: super-observer mission execution.** The Super Observer provides the System with the requested Car Crash Information.
3. **Extension point: super-observer mission executed.** The System adds this information the Car Crash information.
4.2.1.9 UC6b: Execute First-Aid Mission

- **ID**: UC6b
- **Name**: Execute First Aid Mission
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: First-Aid Worker
- **Intention**: The First-Aid Worker executes a First-Aid Mission by offering a the First-Aid Resource.

**Preconditions**:
- The System has successfully requested the First Aid Resource from a First Aid Worker
- First Aid Worker is at the Crisis Location

**Main Success Scenario**:

1. **Extension point: first-aid mission briefing.** The System briefs the First-Aid Worker with the concrete Mission details: the target Victim to rescue, its current Location, as well as the degree of urgency, potential medical history of the Victim, . . . .
2. **Extension point: first-aid mission execution.** The First-Aid Worker rescues the Victim from the Car, offers First-Aid and reports this back to the System.
3. **Extension point: first-aid mission executed.** The System adds this information the Car Crash information.

4.2.1.10 UC6c: Execute Victim Transportation Mission

- **ID**: UC6c
- **Name**: Execute Victim Transportation Mission
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: Helicopter Pilot
- **Intention**: The Helicopter Pilot executes a Victim Transportation Mission by offering a the Victim Transportation Resource.

**Preconditions**:
- The System has successfully requested the Victim Transportation Resource from an Helicopter Pilot

**Main Success Scenario**:

1. **Extension point: victim transportation mission briefing.** The System briefs the Helicopter Pilot with the concrete Mission details: the target Victim to transport, its current Location, as well as the degree of urgency, and the destination Hospital of the Victim, . . . .
2. **Extension point: victim transportation mission execution.** The Helicopter Pilot flies to the Car Crash Location, picks up the Victim and transports it to the Hospital. Finally, she reports this back to the System.
3. **Extension point: victim transportation mission executed.** The System adds this information the Car Crash information.
4.2.1.11 UC7: Update Weather Information

- **ID:** UC9
- **Name:** Update Weather Information
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Weather Station
- **Intention:** The Weather Station updates the CCMS in real-time about changing weather conditions.
- **Preconditions:** The System has indicated interest in Weather Information for a certain location.
- **Main Success Scenario:**
  1. The Weather Station pushes Weather Information updates to the System.
  2. **Extension point:** weather update received. The system receives this information.

4.2.2 Extension use cases

4.2.2.1 EXT_UC1: Check Authentication and Authorization

- **ID:** EXT_UC1
- **Name:** Check Authorization
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** authorization and authentication
- **Extends:**
  - Capture Witness Report (UC3), after witness call routed.
  - Select Strategy (UC4), after coordinator selection.
  - Request Observation (UC5a), after super-observer selection.
  - Request First-Aid (UC5b), after first-aid worker selection.
  - Request Victim Transportation (UC5c), after helicopter pilot selection.
  - Execute Super-Observer Mission (UC6a), before super-observer mission briefing.
  - Execute First-aid Mission (UC6b), before first-aid mission briefing.
  - Execute Victim Transportation Mission (UC6c), before victim transportation mission briefing.
- **Primary Actor:** CMS Employee
- **Intention:** The system checks whether or not the CMS Employee is authenticated and has the access rights for this action.
- **Postconditions:** The check has succeeded (authentication and authorization is ok).
- **Main Success Scenario:**
  1. The System checks whether or not the CMS Employee is authenticated and authorized to perform the requested action *(include: authenticate and authorize (UC2))*
- **Extensions:**
  1b. Authentication and/or authorization check of the CMS Employee failed.
    1. Include authenticate and authorize (UC2).
4.2.2.2 EXT_UC2: Validate Witness Identity

- **ID:** EXT_UC2
- **Name:** Validate Witness Identity
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** witness identity validation
- **Secondary Actor:** Phone Company
- **Extends:**
  - Capture Witness Report (UC3), after witness identity known.
- **Primary Actor:** Coordinator
- **Intention:** The Coordinator asks identity information which is checked in parallel by the Phone Company.
- **Postconditions:** The Witness identity is validated.

**Main Success Scenario:**

1. In parallel to the next steps:
   1. System contacts the Phone Company to verify witness information.
   2. Phone Company sends address/phone information to System.
   3. **Extension point: witness identity validated.** System validates information received from the Phone Company, and the Witness identity is confirmed.

**Extensions:**

1b **Extension point: fake witness.** System validates information received from the Phone Company, and the Witness identity is not confirmed.

1. The use case failed.

4.2.2.3 EXT_UC3: Confirm Car Crash

- **ID:** EXT_UC3
- **Name:** Confirm Car Crash
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** crisis confirmation
- **Extends:**
  - Capture Witness Report (UC3), at car crash location known.
- **Primary Actor:** Coordinator
- **Secondary Actor:** Surveillance System
- **Intention:** The System consults the SurveillanceSystem for live images at the provided location to confirm the car crash
- **Postconditions:** The car crash is confirmed.

**Main Success Scenario:**

1. In parallel to the next steps:
   1. System requests video feed from SurveillanceSystem.
   2. Surveillance System starts sending video feed to System.
   3. System starts displaying video feed for Coordinator.
   4. **Extension point: car crash validated.** Coordinator confirms the crash visually and changes the state of the crisis to ‘confirmed’

**Extensions:**

1.4b **Extension point: fake car crash.** Camera vision of the location is perfect, but Coordinator cannot confirm the situation that the witness describes or the Coordinator determines that the witness is calling in a fake crisis.

1. The use case has failed.
4.2.2.4 EXT_UC4: Update Car Crash Information

- **ID**: EXT_UC4
- **Name**: Update Car Crash Information
- **Scope**: Car Crash Crisis Management System
- **Main Concern**: real-time update
- **Extends**:
  - Update Weather Information (UC7), at weather update received
  - Execute Super-Observer Mission (UC6a), at super-observer mission executed
  - Execute First-Aid Mission (UC6b), at first-aid mission executed
  - Execute Victim Transportation Mission (UC6c), at victim transportation mission executed
- **Primary Actor**: Resource Provider
- **Intention**: The Resource Providers updates the progress, state and/or information regarding the car crash to the system.
- **Preconditions**: The Resource Provider is known in the system.
- **Postconditions**: The Car Crash crisis information is updated.
- **Main Success Scenario**:
  1. The Resource Provider updates the progress, state and/or information regarding the car crash to the system.
  2. The System updates the car crash information.

4.2.2.5 EXT_UC5: Revise Strategy

- **ID**: EXT_UC5
- **Name**: Revise Strategy
- **Scope**: Car Crash Crisis Management System
- **Main Concern**: strategy revision
- **Extends**:
  - Update Weather Information (UC7), at weather update received
  - Update Car Crash Information (EXT_UC4), at car crash information updated
- **Primary Actor**: System
- **Intention**: The System revises the ongoing Strategy as result of changing conditions.
- **Postconditions**: The Strategy is revised.
- **Main Success Scenario**:
  1. The system calculates how the changing conditions affect the current Strategy, considers alternative Strategies (minimizing risk, cost, maximizing safety, timeliness based on urgency, . . .).
  2. **Extension point**: strategy revision. The system proposes a set of revised strategies to the Coordinator (Include: Select Strategy (UC4)).
- **Extensions**:
  2.b The changing conditions does not trigger a strategy revision. Use case ends.
4.2.2.6  EXT_UC6: Check Resource Provider Availability

- **ID:** EXT_UC6
- **Name:** Check Resource Provider Availability
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Resource Provider Availability
- **Extends:**
  - Request Observation (UC5a), after super-observer selection.
  - Request First-Aid (UC5b), after first-aid worker selection.
  - Request Victim Transportation (UC5c), after helicopter pilot selection.
  - Capture Witness Report (UC3), after witness call routed.
  - Select Strategy (UC4), after coordinator selection.
- **Primary Actor:** System
- **Intention:** The System looks up the availability status of a Resource Provider.
- **Preconditions:** The Resource Provider is known in the system.
- **Postconditions:** This use case returns the availability status of the Resource Provider.
- **Main Success Scenario:**
  1. The System looks up the availability status of the requested Resource Provider.

4.2.2.7  EXT_UC7: Set Resource Provider Status Unavailable

- **ID:** EXT_UC7
- **Name:** Set Resource Provider Status Unavailable
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Resource Provider Availability
- **Extends:**
  - Capture Witness Report (UC3), before call initiated.
  - Execute Super-Observation Mission (UC6a), before super-observation mission execution.
  - Execute First-Aid Mission (UC6b), before first-aid mission execution.
  - Execute Victim Transportation Mission (UC6c), before victim transportation mission execution.
  - Select Strategy (UC4), after strategy selection.
- **Primary Actor:** System
- **Intention:** The System sets the availability status of a Resource Provider to “Unavailable”.
- **Preconditions:** The Resource Provider is known in the system.
- **Postconditions:** The availability status of the Resource Provider is set to “Unavailable”.
- **Main Success Scenario:**
  1. The System sets the availability status of the Resource Provider to “Unavailable”.
4.2.2.8 EXT_UC8: Set Resource Provider Status Available

- **ID:** EXT_UC8
- **Name:** Set Resource Provider Status Available
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Resource Provider Availability
- **Extends:**
  - `Execute Super-Observer Mission (UC6a)`, at super-observer mission executed.
  - `Capture Witness Report (UC3)`, before call ended.
  - `Execute First-Aid Mission (UC6b)`, at first-aid mission executed.
  - `Execute Victim Transportation Mission (UC6c)`, at victim transportation mission executed.
  - `Select Strategy (UC4)`, after strategy selected.
- **Primary Actor:** System
- **Intention:** The System sets the availability status of a Resource Provider to “Available”.
- **Preconditions:** The Resource Provider is known in the system.
- **Postconditions:** The availability status of the Resource Provider is set to “Available”.
- **Main Success Scenario:**
  1. The System sets the availability status of the Resource Provider to “Available”.

4.2.2.9 EXT_UC9: Log Action

- **ID:** EXT_UC9
- **Name:** Log Action
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Statistical Logging
- **Extends:**
  - `Authenticate and Authorize (UC2)`, at authentication success, authorization success, authentication failure, authorization failure.
  - `Capture Witness Report (UC3)`, at witness call routed, witness call initiated, witness identity known, car crash location known, witness report complete, witness call ended.
  - `Select Strategy (UC4)`, at coordinator selection, strategy selection.
  - `Request Observation (UC5a)`, at super-observer selection, super-observer agreement, insufficient super-observers.
  - `Request First-Aid (UC5b)`, at first-aid worker selection, first-aid worker agreement, insufficient first-aid workers.
  - `Request Victim Transportation (UC5c)`, at helicopter pilot selection, helicopter pilot agreement, insufficient helicopters.
  - `Execute First-Aid Mission (UC6b)`, after first-aid mission execution, first-aid mission executed.
  - `Execute Victim Transportation Mission (UC6c)`, after victim transportation mission execution, victim transportation mission executed.
  - `Update Weather Information (UC7)`, at weather update received.
  - `Validate Witness Identity (EXT_UC2)`, at witness identity validated, fake witness.
  - `Confirm Car Crash (EXT_UC3)`, at car crash validated, fake car crash.
- **Primary Actor:** System
- **Intention:** The System writes an entry to the log.
- **Preconditions:** A certain action has triggered this use case.
- **Postconditions:** The log is updated.
- **Main Success Scenario:**
  1. The System writes an entry to the log.
4.2.3 Use case diagram

This activity of separating concerns results in a set of use cases in which the crosscutting concerns are well-modularized: there is less scattering and tangling of the concerns over the use cases. However, this comes at the cost of a very large coupling (originating from the fact that we have separated crosscutting concerns). Figure 16 presents the use case diagram at this point — without the log action (EXT_UC9) use case. Preparing the full diagram at this point becomes very difficult as we have introduced 36 extension points and 63 extend relations. Figure 16 only depicts 18 of these extend relation and already becomes difficult to read. The main purpose of this diagram is to illustrate how unmanageable this model has become after separation.

Figure 16: Use Case Diagram - after separation of crosscutting concerns. The extension use case log action (EXT_UC9) and its massive amount of extension relationships to base and other extension use cases is not shown.

5 Activity 3. Generalize crosscutting extend-relationships

This activity is the heart of the method. To address the problem of high coupling between base and aspect use cases, we introduce a use case inheritance hierarchy, with the goal of generalizing the extension
points as well. This generalization forces the requirements engineer to consult or revisit the analysis of the problem domain and come up with suitable abstractions.

As presented in [5], an algorithm is executed by the requirements engineer to guide this process. The main idea of this algorithm is that we iterate over all extension use case, and then consider for all use cases extended by these use cases whether or not there is an essential commonality. The main question driving this commonality identification process is: “what do use case A and use case B have in common, so they are both extended by the same aspect use case A?”.

This Section does not provide intermediate results of the algorithm, but the end result.

5.1 Use cases

5.1.1 Base use cases

5.1.1.1 UC1: Indicate Availability State

UNCHANGED

- **ID:** UC1
- **Name:** Indicate Availability State
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The CMS Employee indicates his Availability State to the system.
- **Preconditions:** The CMS Employee is authenticated.
- **Postconditions:** The System has stored the Availability State of the CMS Employee and will take it into account when assigning the CMS Employee.
- **Main Success Scenario:**

  1. The CMS Employee indicates he wants to change his Availability State.
  2. The System presents a list of possible Availability States.
  3. The CMS Employee indicates his current Availability State.
  4. The System changes the Availability State of the CMS Employee.

5.1.1.2 UC2: Authenticate and Authorize

UNCHANGED

- **ID:** UC2
- **Name:** Authenticate and Authorize
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The System wants to ensure identity of the CMS Employee and check his access rights.
- **Preconditions:** An CMS Employee has attempted to perform a certain action.
- **Postconditions:** The CMS Employee is authenticated and authorized to perform the requested action.
- **Main Success Scenario:**

  1. If the CMS Employee has not authenticated himself in the last 30 minutes,

     1. the System requests the CMS Employee to provide his credentials (e.g. uname/pwd)
     2. The CMS Employee provides his credentials
     3. **Extension point:** authentication success. The System validates the credentials

     2. **Extension point:** authorization success. The System validates the access rights of the CMS Employee for the attempted action and grants the CMS Employee access.

- **Extensions:**

  1.3b The provided credentials are incorrect
1 Repeat steps 1 and 2 three times.
2 Extension point: authentication failure. Use case fails.
2b The authenticated CMS Employee does not have the necessary access rights to perform the requested action.
1 Extension point: authorization failure. The use case fails, as the requested action is declined.

5.1.1.3 UCa: Perform Crisis Management Action (abstract)
- **ID:** UCa
- **Name:** Perform Crisis Management Action
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The CMS Employee performs an action that can be classified under the term crisis management.
- **Postconditions:**
  - The action was performed.

**Main Success Scenario:**
1. Extension point: action initiation. (Abstract) The System initiates the crisis management action by providing the CMS Employee with all details.
2. Extension point: action performed. (Abstract) The CMS Employee performs the crisis management action successfully and reports this to the system.

**Extensions:**
2.b. Extension point: action failed. (Abstract) The CMS performs the crisis management action but fails, and reports this to the system.

5.1.1.4 UCb: Perform Action Requiring CMS Employee Selection (abstract)
- **ID:** UCb
- **Name:** Perform Action Requiring CMS Employee Selection (abstract)
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** The System triggers an action that requires the selection of a suitable CMS Employee.
- **Postconditions:**
  - A suitable CMS Employee is selected, and the action performed.

**Main Success Scenario:**
1. (Abstract) The System determines that an action is required for which a suitable CMS Employee is to be selected.
2. Extension point: employee selection. The System selects a suitable CMS Employee (in terms of expertise, availability, etc) for the task. The systems asks the CMS Employee if he agrees to perform this action.
3. Extension point: employee agreement. The CMS Employee agrees to perform the action.

**Extensions:**
3.b. The CMS Employee refuses the action.
3.b.+1 Retry (step 1).
  - Extension point: no employee available. If no suitable CMS Employee agreeing to perform the action is found, the use case fails.
5.1.1.5 UC3: Capture Witness Report

- **ID:** UC3
- **Name:** Capture Witness Report
- **Implements:** Perform Crisis Management Action (UCa), and Perform Action Requiring CMS Employee Selection (UCb).
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Coordinator
- **Intention:** The Coordinator intends to create a crisis record based on the information obtained from the witness.

**Postconditions:**
- A car crash has been reported and is known to the system. An emergency level has been assigned.
- The Witness Report is recorded.
- The System has subscribed to Weather information for the Car Crash location.

**Main Success Scenario:**

1. \((\text{Implements UCb, step 1}): \text{Witness calls in to report a car crash}\)
2. As parent UCb, step 2.
3. As parent UCb, step 3.
4. The following steps specialize parent UCb, step 3:step:ucb:action:
   - \((\text{Implements parent UCa, step 1}). \text{The system forwards the witness call to the Coordinator}\)
   - The following steps specialize parent UCa, step 2:
     1. The Coordinator answers the call
     2. The System starts recording the call
     3. Coordinator requests the Witness to provide his identification information. (Witness information includes the first name, last name, phone number, and address.)
     4. The Witness provides the requested identification information.
   5. **Extension point:** witness identity known. Coordinator provides the identification information as reported by the witness to the System.
   6. The Coordinator requests the witness to provide the location of the car crash
   7. The Witness provides the location information
   8. **Extension point:** car crash location known. The Witness provides the System of location and type of crisis as reported by the witness.
   9. System subscribes to Weather Information for the Car Crash Location by contacting the appropriate Weather Station.
   10. System provides Coordinator with a crisis-focused checklist to adhere to.
   11. Coordinator requests additional car crash information (e.g. number, types of Vehicles involved, vehicle identification)
   12. Witness provides the additional car crash information.
   13. Coordinator provides crisis information to System as reported by the witness.
   14. System assigns an initial emergency level to the crisis and sets the crisis status to active.
   15. **Extension point:** witness call ended. The system ends the call.
   16. The system stops recording.

**Extensions:**

- 3b. As in parent UCb, steps 3.b and following.
  - The following alternatives implement parent UCa, step 2.b:
    \{4.1,4.2,4.3,4.4,4.5,4.6,4.7,4.8\}b Call is disconnected.
    i. The system ignores the witness report (insufficient information)
ii. The system stops the recording.

\{4.9,4.10,4.11.14.12\}b The call is disconnected.

i. The system attempts to re-dial the Witness.

ii. Witness answers the call.

iii. Systems continues the use case.

\{4.9,4.10,4.11.14.12\}c The call is disconnected.

i. The system attempts to re-dial the Witness.

ii. Witness does not answer the call.

iii. Systems continues the use case at step 17.

5.1.1.6 UC4: Select Strategy

- **ID**: UC4
- **Name**: Select Strategy
- **Implements**: Perform Crisis Management Action (UCa).
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: Coordinator
- **Intention**: The System manages the car crash by suggesting a number of alternative scenarios to the Coordinator.
- **Preconditions**:
  - A car crash was reported to the system.
  - The system has calculated a (number of) strategy/strategies to deal with this crisis.
- **Postconditions**: A strategy has been selected and the system will follow it.
- **Main Success Scenario**:
  1. *(Implements UCa, step 1)*: The System searches for a Coordinator (based on expertise and availability)
  2. The following steps specialize parent UCa, step 3:step:uca:success:
     - The System presents the Coordinator with the alternative Strategies and asks him to select one.
     - Coordinator selects one Strategy.
- **Extensions**:
  1b. *(Implements parent UCa, steps 2.b)* No suitable Coordinator is found. The use case fails.

5.1.1.7 UC5: Request Resource (abstract)

- **ID**: UC5
- **Name**: Request Resource *(abstract)*
- **Implements**: Perform Action Requiring CMS Employee Selection (UCb).
- **Scope**: Car Crash Crisis Management System
- **Primary Actor**: Resource Provider
- **Intention**: The currently executing Mission *(abstract)* requires some Resource *(abstract)*. Therefore, a Resource Provider *(abstract)* that has the expertise to offer the Resource *(abstract)* is contacted.
- **Preconditions**: The System is following a certain Strategy which consists of a Mission *(abstract)* requiring the requested Resource *(abstract)*.
- **Main Success Scenario**:
  1. *(Overrides UCb, step 1), but still abstract*: The system triggers the executes of a Mission which requires a certain Resource provided by a Resource Provider.
  2. As parent UCb, step 2.
  3. As parent UCb, step 3.
  4. The following step specializes parent UCb, step 3:step:ucb:action:
     1 The use case succeeds.
- **Extensions**:
  3b. As in parent UCb, steps 3.b and following.
5.1.1.8 UC5a: Request Observation

- **ID:** UC5a
- **Name:** Request Observation
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Super Observer
- **Intention:** The currently executing Super-Observation Mission requires some Observation as a Resource. Therefore, a Super-Observer that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a Super-Observer Mission requiring the requested Observation Resource.
- **Main Success Scenario:**
  1. (Implements parent use case, step 1). The Strategy currently being followed requires the “Observation” resource.
  2. Other steps, as parent.
- **Extensions:** as parent.

5.1.1.9 UC5b: Request First-Aid

- **ID:** UC5b
- **Name:** Request First-Aid (Implements UC5)
- **Scope:** Car Crash Crisis Management
- **Primary Actor:** First-Aid Worker
- **Intention:** The currently executing First-Aid Mission requires some First-Aid as a Resource. Therefore, a First-Aid Worker that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a First-Aid Mission requiring the requested First-Aid Resource.
- **Main Success Scenario:**
  1. (Implements parent use case, step 1). The Strategy currently being followed requires the “First-Aid” resource.
  2. Other steps, as parent.
- **Extensions:** as parent.

5.1.1.10 UC5c: Request Victim Transportation

- **ID:** UC5c
- **Name:** Request Victim Transportation (Implements UC5)
- **Scope:** Car Crash Crisis Management
- **Primary Actor:** Helicopter
- **Intention:** The currently executing Victim Transportation Mission requires some Victim Transportation as a Resource. Therefore, a Helicopter that has the necessary expertise is contacted.
- **Preconditions:** The System is following a certain Strategy which consists of a Victim Transportation Mission requiring the requested Victim Transportation Resource.
- **Main Success Scenario:**
  1. (Implements parent use case, step 1). The Strategy currently being followed requires the “Victim Transportation” resource.
  2. Other steps, as parent.
- **Extensions:** as parent.
5.1.1.11 UCc: Send Car Crash Information Update (abstract)

- **ID:** UCb
- **Name:** Send Car Crash Information Update (abstract)
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** CMS Employee
- **Intention:** This abstract use case covers the functionality of sending back to the system updates regarding the state of the car crash.
- **Postconditions:**
  - The system has updated its information regarding the car crash situation.

**Main Success Scenario:**
1. *(Abstract)* The main actor sends an information update regarding the car crash.
2. **Extension point:** update received. The System receives the update and updates the car crash information.

5.1.1.12 UC6: Execute Mission (abstract)

- **ID:** UC6
- **Name:** Execute Mission *(abstract)*
- **Specializes:** Send Car Crash Information Update *(UCc)* and Perform Crisis Management Action *(UCa)*
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Resource Provider
- **Intention:** The Resource Provider executes a Mission *(abstract)* by offering a certain Resource *(abstract)*.
- **Preconditions:** The System has successfully requested a Resource *(abstract)* from a Resource Provider *(abstract)*.
- **Main Success Scenario:**
  1. *(Specializes parent UCa, step 1)* *(Abstract)* The System briefs the Resource Provider with the Mission-specific details.
  2. **Implements both parent UCa, step 2 and parent UCc, step 1.** *(Abstract)* The Resource Provider executes the Mission *(abstract)* and reports back to the System.

**Extensions:**
2b. *(Specialize parent UCa, step 2.b.)* *(Abstract).* The Mission failed.

5.1.1.13 UC6a: Execute Super Observer Mission

- **ID:** UC6a
- **Name:** Execute Super Observer Mission
- **Specializes:** Execute Mission *(UC6)*
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Super Observer
- **Intention:** The Super Observer executes a Super Observer Mission by offering the Observation Resource.
- **Preconditions:**
  - The System has successfully requested the Observation Resource from a Super Observer
  - Super Observer is at the Crisis Location.

**Main Success Scenario:**
1. *(Implements parent UC6, step 1).* The System requests the Super Observer to provide specific information about the Car Crash.

2. *(Implements parent UC6, step 2).* The Super-Observer provides the specific information to the System.

- **Extensions:**

2b. *(Implement parent UC6, step 2b.)* The Super Observer did not manage to execute this mission. Use case fails.

---

### 5.1.1.14 UC6b: Execute First-Aid Mission

- **ID:** UC6b
- **Name:** Execute First Aid Mission
- **Specializes:** Execute Mission (UC6)
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** First Aid Worker
- **Intention:** The First Aid Worker executes a First Aid Mission by offering a the First Aid Resource.

**Preconditions:**
- The System has successfully requested the First Aid Resource from a First Aid Worker
- First Aid Worker is at the Crisis Location.

- **Main Success Scenario:**

1. *(Implements parent UC6, step 1).* The System briefs the First Aid Worker with the concrete Mission details: the target Victim to rescue, its current Location, as well as the degree of urgency, potential medical history of the Victim, . . . .

2. *(Implements parent UC6, step 2).* The First-Aid Worker offers First-Aid to the Victim and reports this back to the System.

- **Extensions:**

2b. *(Implement parent UC6, step 2b.)* The First-Aid Worker did not manage to execute this mission. Use case fails.

---

### 5.1.1.15 UC6c: Execute Victim Transportation Mission

- **ID:** UC6b
- **Name:** Execute First Aid Mission *(Implements UC6)*
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Helicopter Pilot
- **Intention:** The Helicopter Pilot executes a Victim Transportation Mission by offering a the Victim Transportation Resource.

**Preconditions:**
- The System has successfully requested the Victim Transportation Resource from an Helicopter.

- **Main Success Scenario:**

1. *(Implements parent UC6, step 1).* The System briefs the Helicopter Pilot with the concrete Mission details: the target Victim to transport, its current Location, as well as the degree of urgency, the destination Hospital of the Victim, . . . .

2. *(Implements parent UC6, step 2).* The Helicopter Pilot picks up the Victim and transports him to the Hospital. Finally, she reports this back to the System.

- **Extensions:**

2b. *(Implement parent UC6, step 2b.)* The Helicopter Pilot did not manage to execute this mission. Use case fails.
5.1.1.16 UC7: Update Weather Information

- **ID:** UC9
- **Name:** Update Weather Information
- **Implements:** Send Car Crash Information Update (UCc)
- **Scope:** Car Crash Crisis Management System
- **Primary Actor:** Weather Station
- **Intention:** The Weather Station updates the CCMS in real-time about changing weather conditions.
- **Preconditions:** The System has indicated interest in Weather Information for a certain location.
- **Main Success Scenario:**
  1. (Implements parent UCc, at step 1.) The Weather Station pushes Weather Information updates to the System.

5.1.2 Extension use cases

5.1.2.1 EXT_UC1: Check Authentication and Authorization

- **ID:** EXT_UC1
- **Name:** Check Authorization
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** authorization and authentication
- **Extends:**
  - Perform a crisis management action (UCa), at action initiation.
- **Primary Actor:** CMS Employee
- **Intention:** The system checks whether or not the CMS Employee is authenticated and has the access rights for this action.
- **Postconditions:** The check has succeeded (authentication and authorization is ok).
- **Main Success Scenario:**
  1. The System checks whether or not the CMS Employee is authenticated and authorized to perform the requested action (include: authenticate and authorize (UC2))
- **Extensions:**
  1b. Authentication and/or authorization check of the CMS Employee failed.
       1. Include authenticate and authorize (UC2).

5.1.2.2 EXT_UC2: Validate Witness Identity

- **ID:** EXT_UC2
- **Name:** Validate Witness Identity
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** witness identity validation
- **Secondary Actor:** Phone Company
- **Extends:**
  - Capture Witness Report (UC3), after witness identity known.
- **Primary Actor:** Coordinator
- **Intention:** The Coordinator asks identity information which is checked in parallel by the Phone Company.
- **Postconditions:** The Witness identity is validated.
- **Main Success Scenario:**
  
  1. In Parallel to the next steps:
     1. System contacts the Phone Company to verify witness information.
     2. Phone Company sends address/phone information to System.
     3. **Extension point:** witness identity validated. System validates information received from the Phone Company, and the Witness identity is confirmed.

- **Extensions:**
  
  1.3b **Extension point:** fake witness. System validates information received from the Phone Company, and the Witness identity is not confirmed.
  1. The use case failed.

### 5.1.2.3 EXT_UC3: Confirm Car Crash

- **ID:** EXT_UC3
- **Name:** Confirm Car Crash
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** crisis confirmation
- **Extends:**
  - Capture Witness Report (UC3), at car crash location known.
- **Primary Actor:** Coordinator
- **Secondary Actor:** Surveillance System
- **Intention:** The System consults the Surveillance System for live images at the provided location to confirm the car crash
- **Postconditions:** The car crash is confirmed.

- **Main Success Scenario:**
  
  1. In parallel to the next steps:
     1. System requests video feed from Surveillance System.
     2. Surveillance System starts sending video feed to System.
     3. System starts displaying video feed for Coordinator.
     4. **Extension point:** car crash validated. Coordinator confirms the crash visually and changes the state of the crisis to ‘confirmed’

- **Extensions:**
  
  1.4b **Extension point:** fake car crash. Camera vision of the location is perfect, but Coordinator cannot confirm the situation that the witness describes or the Coordinator determines that the witness is calling in a fake crisis.
  1. The use case has failed.

### 5.1.2.4 EXT_UC4: Update Car Crash Information

- **ID:** EXT_UC4
- **Name:** Update Car Crash Information
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** real-time update
- **Extends:**
  - Send Car Crash Information Update (UCc), at update received.
- **Primary Actor:** Resource Provider
**Intention:** The Resource Providers updates the progress, state and/or information regarding the car crash to the system.

**Preconditions:** The Resource Provider is known in the system.

**Postconditions:** The Car Crash crisis information is updated.

**Main Success Scenario:**

1. The Resource Provider updates the progress, state and/or information regarding the car crash to the system.
2. The System updates the car crash information.

### 5.1.2.5 EXT_UC5: Revise Strategy

**ID:** EXT_UC5  
**Name:** Revise Strategy  
**Scope:** Car Crash Crisis Management System  
**Main Concern:** strategy revision  
**Extends:**  
- Send Car Crash Information Update (UCc), at update received.

**Primary Actor:** System  
**Intention:** The System revises the ongoing Strategy as result of changing conditions.  
**Postconditions:** The Strategy is revised.

**Main Success Scenario:**

1. The system calculates how the changing conditions affect the current Strategy, considers alternative Strategies (minimizing risk, cost, maximizing safety, timeliness based on urgency, ...).
2. **Extension point:** strategy revision. The system proposes a set of revised strategies to the Coordinator (Include: Select Strategy (UC4)).

**Extensions:**

2.b The changing conditions does not trigger a strategy revision. Use case ends.

### 5.1.2.6 EXT_UC6: Check Resource Provider Availability

**ID:** EXT_UC6  
**Name:** Check Resource Provider Availability  
**Scope:** Car Crash Crisis Management System  
**Main Concern:** Resource Provider Availability  
**Extends:**  
- Perform Action Requiring CMS Employee Selection (UCb), at employee selection.

**Primary Actor:** System  
**Intention:** The System looks up the availability status of a Resource Provider.  
**Preconditions:** The Resource Provider is known in the system.  
**Postconditions:** This use case returns the availability status of the Resource Provider.

**Main Success Scenario:**

1. The System looks up the availability status of the requested Resource Provider.
5.1.2.7 EXT_UC7: Set Resource Provider Status Unavailable
- **ID:** EXT_UC7
- **Name:** Set Resource Provider Status Unavailable
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Resource Provider Availability
- **Extends:**
  - Perform a crisis management action (UCa), at action initiation.
- **Primary Actor:** System
- **Intention:** The System sets the availability status of a Resource Provider to “Unavailable”.
- **Preconditions:** The Resource Provider is known in the system.
- **Postconditions:** The availability status of the Resource Provider is set to “Unavailable”.
- **Main Success Scenario:**
  1. The System sets the availability status of the Resource Provider to “Unavailable”.

5.1.2.8 EXT_UC8: Set Resource Provider Status Available
- **ID:** EXT_UC8
- **Name:** Set Resource Provider Status Available
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Resource Provider Availability
- **Extends:**
  - Perform a crisis management action (UCa), at action performed.
- **Primary Actor:** System
- **Intention:** The System sets the availability status of a Resource Provider to “Available”.
- **Preconditions:** The Resource Provider is known in the system.
- **Postconditions:** The availability status of the Resource Provider is set to “Available”.
- **Main Success Scenario:**
  1. The System sets the availability status of the Resource Provider to “Available”.

5.1.2.9 EXT_UC9: Log Action
- **ID:** EXT_UC9
- **Name:** Log Action
- **Scope:** Car Crash Crisis Management System
- **Main Concern:** Statistical Logging
- **Extends:**
  - Perform a crisis management action (UCa), at action initiated, action performed, action failed.
  - Perform Action Requiring CMS Employee Selection (UCb), at employee selection, employee agreement, and no employee available.
  - Send Car Crash Information Update (UCc), at update received.
  - Indicate Availability (UC1), at availability information stored.
  - Authenticate and Authorize (UC2), at authentication success, authorization success, authentication failure, authorization failure.
  - Select Strategy (UC4), at strategy selection.
  - Validate Witness Identity (EXT_UC2), at witness identity validated, fake witness.
  - Confirm Car Crash (EXT_UC3), at car crash validated, fake car crash.
• Primary Actor: System
• Intention: The System writes an entry to the log.
• Preconditions: A certain action has triggered this use case.
• Postconditions: The log is updated.
• Main Success Scenario:
  1. The System writes an entry to the log.

5.2 Use case diagram

![Use Case Diagram](image)

Figure 17: Use Case Diagram - after generalization (activity 3 of the method).

In summary, we have reduced a total of 36 extension points in activity 2 to a total of 18 extension points. The number of extension relationships has been reduced from 63 in activity 2 to 16.
6 Activity 4. Define Stable Domain Abstractions

6.1 Domain Abstraction Inventory

The following 18 domain abstractions are defined, one for each extension point. Contextual information is underlined.

6.1.1 car crash information update received (short: update received):

This domain abstraction denotes that

- a source of Car Crash information (either an External Resource Provider or an External Information Provider) has presented the system with
- an update (new information) regarding the car crash.

**Link to Domain Models:** External Resource Providers are presented in Figure 9, while External Information Providers are presented in Figure 10. However, the fact that External Resource Providers or External Information Providers continually update the system’s view on the car crash was not prominent in the domain models.

6.1.2 authentication success:

This domain abstraction denotes successful authentication of

- a CMS Employee.

**Link to Domain Models:** Authentication was presented in Figure 8.

6.1.3 authentication failed:

This domain abstraction denotes authentication failure of

- a CMS Employee.

**Link to Domain Models:** Authentication was presented in Figure 8.

6.1.4 authorization success:

This domain abstraction denotes successful authorization of

- a CMS Employee.

**Link to Domain Models:** Authorization was presented in Figure 8.

6.1.5 authorization failed:

This domain abstraction denotes authorization failure of

- a CMS Employee.

**Link to Domain Models:** Authorization was presented in Figure 8.

6.1.6 witness identity known:

This domain abstraction denotes the fact that the System has successfully received

- a Witness Identity information. This is usually a name and a phone number.

**Link to Domain Models:** Witnesses, their identity and the validation of their identity are concepts presented in Figure 5.
6.1.7 witness identity validated:
This domain abstraction denotes the fact that
• a Witness identity validation was successful.

Link to Domain Models: Witnesses, their identity and the validation of their identity are concepts presented in Figure 5.

6.1.8 fake witness:
This domain abstraction denotes the fact that
• a Witness identity validation failed and the witness is considered to be an imposter.

Link to Domain Models: Witnesses, their identity and the validation of their identity are concepts presented in Figure 5.

6.1.9 car crash location known
This domain abstraction denotes the fact that the System has successfully received
• the location of the car crash.

Link to Domain Models: The location of a car crash and the visual confirmation of a car crash is modeled in Figure 5.

6.1.10 car crash confirmed:
This domain abstraction denotes that the location-based confirmation of
• a car crash was successful.

Link to Domain Models: The location of a car crash and the visual confirmation of a car crash is modeled in Figure 5.

6.1.11 fake car crash:
This domain abstraction denotes that the location-based confirmation of
• a car crash failed, and therefore the Witness report should not be taken seriously.

Link to Domain Models: The location of a car crash and the visual confirmation of a car crash is modeled in Figure 5.

6.1.12 strategy selected:
This domain abstraction denotes the selection of a
• Strategy, by a
• Coordinator.

Link to Domain Models: The existence of multiple strategies, and the distinguishing factors such as risk and cost are modeled in Figure 5. However, the actual selection between alternative strategies by the Coordinator was not presented in the domain models.

6.1.13 employee selection:
This domain abstraction denotes the selection of
• CMS Employee
• for a certain action.
6.1.14 employee agreement:
This domain abstraction denotes the explicit agreement of

- a CMS Employee
- to perform a certain action.

6.1.15 no employee available:
This domain abstraction denotes the lack of a suitable CMS Employee

- to perform a certain action.

6.1.16 car crash management action initiation (short: action initiation):
This domain abstraction denotes the initiation of a

- car crash management action, by a
  - Resource Provider or a CMS Employee.

6.1.17 car crash management action performed (short: action performed):
This domain abstraction denotes the successful execution of a

- car crash management action, by a
  - Resource Provider or a CMS Employee.

6.1.18 car crash management action failed (short: action failed):
This domain abstraction denotes the failure of a

- car crash management action, by a
  - Resource Provider or a CMS Employee.
6.2 Updated domain models

A side-effect of the application of the approach is that we now have a more in-depth understanding of the problem domain and we came up with several domain abstraction that may or may not have been part of the domain analysis of Section 3. It is very worthwhile at this point to iterate once more over the actual domain models and fill in the gaps identified during this activity of the method. We use the color ‘red’ to illustrate what was is adapted when compared to the domain models of Section 3.2.2.

![Domain Model: Car Crash Resolution](image)

Figure 18: Domain Model: Car Crash Resolution
Figure 19: Domain Model: External Information Providers

Figure 20: Domain Model: Crisis Management Action
7 Activities 5–7. Architecture Design for Reusable Pointcut Interfaces

7.1 Activity 5: Initial Architecture Design

7.1.1 Client-Server view

![Initial architecture diagram]

Figure 21: Initial architecture.

7.1.2 Key Architectural Decisions

Below, we document the key architectural decisions. Note that none of these are imposed by the method to design reusable pointcut interfaces, as our method only constrains the architecture to ensure compatibility with the proposed stable domain abstraction. At the worst, this involves adapting interfaces slightly to expose additional contextual information. Beyond that, the method has no further impact on the decomposition of the architecture.
• **Multi-tier architectural pattern.** As the system will deal with a wide variety of (i) users, (iii) mobile devices, (iii) communication protocols of these users with the system, we have adopted a *multi-tier* architecture to separate the key business logic (managing crises) from presentation logic. There is (1) the client tier – the software to be deployed on a mobile device, (2) the client facade layer – responsible for separating client presentation logic and business logic, (3) the business tier – consisting of the key decision-making software in crisis management and (4) the data tier – handling persistence of data. Evidently, this architectural decision had a large impact on the architecture, as it very often introduces intermediary interface passing messages across tiers (e.g. a message from the client tier to the business tier *must* pass through the client facade tier.

• **Thin clients, but not web browsers.** It is instructive that the System has the means to update the clients (e.g. Resource Workers or the Coordinator) in real-time. Therefore, adopting an active polling strategy to keep these clients up-to-date for would not be sufficient, and would introduce communication overhead (conflicts with the *mobility* concern). Therefore, many of the client components have explicit call-back interfaces, used by the system to push information to them in real-time. Examples are the *strategy selection* interface of the Coordinator Workstation component, and the *brief* interface of the External Resource Provider component.

### 7.1.3 Components and Interfaces: client tier

The client-server view of the architecture defines the following components and interfaces:

#### 7.1.3.1 Coordinator Workstation.

This is a client, to be used by the Coordinator. It depends fully on the Coordinator Facade component.

  - **strategy selection.** This interface is offered to the System to notify the Coordinator of several suitable strategies and to allow him to select a strategy.
    - selectStrategy(Strategy[]):Strategy

  - **witness call.** This interface is used by the System to forward witness phone calls to.
    - answerCall(Call)

#### 7.1.3.2 External Information Provider.

This is a client, to be used by the External Information Provider. It depends fully on the External Information Provider Facade component. Examples of External Information Providers are the Weather Station, the Phone Company or the Surveillance System. An External Information Provider can either be accessed synchronously (by calling the consult interface), or asynchronously through a subscribe-callback mechanism.

  - **subscribe.** This interface is offered to the System to subscribe to External Information that might be relevant to the System (e.g. weather information).
    - subscribeToInformation(Query)

  - **consult.** This interface is offered to the System to synchronously consult External Information that might be relevant to the System (e.g. checking a phone number, or fetching a traffic camera image feed.)
    - consultInformation(Query)

#### 7.1.3.3 External Resource Provider.

This is a client, to be used by the External Resource Provider. It depends fully on the External Resource Provider Facade component. This is a very thin client, as it is supposed to run on mobile devices.
brief. This interface is offered to the System to brief the Resource Provider with the Mission-specific details.

- brief(Mission)

7.1.4 Components and Interfaces: client facade tier

7.1.4.1 Coordinator Facade.

This is the front-end component, to be used by the Coordinator Workstation. It offers the following interfaces:

auth. Used to authenticate the Coordinator.

- authenticate(UserId,Credentials):Token

strategy selection. This interface is offered to the System to notify the Coordinator of several suitable strategies and to allow him to select a strategy.

- selectStrategy(Strategy[]):Strategy

witness report. This interface is used by the Coordinator to file a witness report.

- initiateWitnessReport(Token,Call):ReportId
- enterWitnessIdentity(Token,ReportId,Name|SSID)
- enterCrashLocation(Token,ReportId,Location)
- enterAdditionalCarCrashInformation(Token,ReportId,Information[])

av. This interface is used by the Coordinator to indicate his availability state.

- setAvailabilityStatus(Token,AvailabilityStatus)

call fwd. This interface is used by the System to forward incoming witness calls to a Coordinator.

- forwardCall(Coordinator,Call)

7.1.4.2 External Information Provider Facade.

This is the front-end component, to be used by the External Information Provider. It offers the following interface:

consult. This interface is offered to consult External Information that might be relevant to the System (e.g. checking a phone number, or fetching a traffic camera image feed.) Depending on the type of information requested and the External Information Provider, this might be done synchronously or asynchronously.

- consultInformation(Query)

update. This callback interface is offered to the External Resource Provider to update the system with External Information when the System has subscribed to it (e.g. a weather information update). Also, it provides a method to trigger re-evaluation and re-calculation of the most suitable strategy.

- updateInformation(Token,Information)
- recalculateStrategies(CarCrashId c)
7.1.4.3 External Resource Provider Facade.

This is the front-end component, to be used by the External Resource Provider. It offers the following interface:

auth. Used to authenticate the External Resource Provider.
- authenticate(UserId,Credentials):Token

av. This interface is used by the External Resource Provider to indicate his availability state.
- setAvailabilityStatus(Token,AvailabilityStatus)

update. This callback interface is offered to the External Resource Provider to update the system with External Information when the System has subscribed to it (e.g. a weather information update). Also, it provides a method to trigger re-evaluation and re-calculation of the most suitable strategy.
- updateInformation(Token,Information)
- recalculateStrategies(CarCrashId c)

brief. This interface is offered to the System to brief the Resource Provider with the Mission-specific details.
- brief(ResourceProvider,Mission)

7.1.5 Components and Interfaces: business tier

7.1.5.1 Authentication and Authorization.

This component is used to authenticate and authorize the user. It checks credentials and if correct, produces a token to be used throughout the system. It offers the following interface:

auth. Used to authenticate a CMS Employee.
- authenticate(UserId,Credentials):Token

7.1.5.2 Authentication and Authorization Check.

This component is used to check a token for validity. Enclosed in the token is authorization information. Therefore, the token suffices to check both authentication and authorization for a CMS Employee.

check. Used to check a token offered by a CMS Employee.
- checkToken(token,Action):boolean

7.1.5.3 Logger.

This component is used to write entries to the persistent Log (see Log DB)

log. Used to write entries to the log.
- log(entry):void

7.1.5.4 Identity Validation.

This component contains the logic to validate the identity of a Witness. This involves consulting a trusted third-party External Resource Provider via the front-end component, External Resource Provider Facade.
check id. User to validate the identity of a Witness.

- validateIdentity(PhoneNumber,Name|SSid):boolean

7.1.5.5 Car Crash Confirmation.

This component contains the logic to confirm a car crash, or to increase confidence the System should have in the witness report. This involves consulting one or more trusted third-parties External Information Provider via the front-end component, External Information Provider Facade.

confirm. Used to visually confirm a car crash on a certain location.

- confirmCarCrash(Location):boolean

7.1.5.6 CMS Employee Management.

This component is used to maintain the CMS Employee information, most relevant of which is the availability state.

availability state. Used to change, set or consult the Availability State of a certain CMS Employee.

- setAvailabilityState(CMSEmployee,AvailabilityState):void
- getAvailabilityState(CMSEmployee):AvailabilityState

7.1.5.7 Car Crash Management.

This central component contains the business logic to manage a car crash. Also, this is the component where witness calls are forwarded to from the Phone System component. The system then selects a suitable employee and forwards the call to the suitable Coordinator Workstation via the Coordinator Facade component.

witness report. This interface is used by the Coordinator to file a witness report.

- initiateWitnessReport(Coordinator,Call):ReportId
- enterWitnessIdentity(ReportId,Name|SSID)
- enterCrashLocation(ReportId,Location)
- enterAdditionalCarCrashInformation(ReportId,Information[])
- finishWitnessReport(Coordinator,ReportId)

call fwd. This interface is used by the Phone System to forward incoming witness calls to the Car Crash Management component, which then selects a suitable Coordinator to handle the call.

- forwardCall(Coordinator,Call)

update. Used to update a car crash with new or additional information.

- update(CarCrashId,Information):void
- recalculateStrategies():void

7.1.5.8 Phone System.

This component is where incoming witness calls arrive. By default (or by configuration), the Phone System routes these calls to the Car Crash Management component.
7.1.5.9 Mission Execution.

This component drives the execution of a Mission, handles interactions with the External Resource Provider via the External Resource Provider Facade. This component is also responsible for selecting a suitable Resource Provider for the required Resource involved in a Mission.

- update. Used to update a car crash with new or additional information.
  - update(CarCrashId, Information):void

- trigfinishWitnessReportger. Used to start the execution of a Mission and invoke Resource requests to Resource Providers.
  - trigger(Mission):void

- accept mission. This interface is offered to the Resource Provider Facade to indicate acceptance of a mission for a certain Resource Provider.
  - accept(ResourceProvider, Mission)

7.1.6 Components and Interfaces: data tier

7.1.6.1 Car Crash DB.
CRUDS interface

7.1.6.2 CMS Employee DB.
CRUDS interface

7.1.6.3 Log DB.
CRUDS interface
7.2 Activity 6: Design Pointcut Interfaces

7.2.1 Design of Pointcut Interfaces

In this architecture, we have defined 19 distinct pointcut interfaces. Figure 22 illustrates which components are designed to expose a certain pointcut interface. Below, we adopt the AspectJ syntax to define these pointcut interfaces.

- **Pointcut interfaces are mostly in the business tier.** This is a result of the method. As the style adopted in the use cases tends to focus on business logic, not on presentation, or persistence logic. The identified domain abstractions in these use cases are specific to the car crash management business-logic. This results in the pointcut interfaces implemented fully in the business tier.

Figure 22: Initial architecture, extended with the pointcut interfaces (depicted as a grey 'lollipop' interface).
7.2.1.1 update received.

public class UpdateReceived{
    private pointcut updateReceivedERP(CarCrashId c, ExternalResourceProvider p, Information i):
        call( IUpdate.update(c,p,i) && args(c,p,i) )
    private pointcut updateReceivedEIP(CarCrashId c, ExternalInformationProvider p, Information i):
        call( IUpdate.update(c,p,i) && args(c,p,i) )
    public pointcut updateReceived(CarCrashId c, Person p, Information i):
        updateReceivedERP(c,p,i) ||
        updateReceivedEIP(c,p,i)
}

7.2.1.2 authentication success.

public class AuthSuccess{
    public pointcut authSuccess(CMSEmployee c):
        call( IAuth.authenticate(c,creds):result && args(c) && (result==true) )
}

7.2.1.3 authentication failure.

public class AuthSuccess{
    public pointcut authSuccess(CMSEmployee c):
        call( IAuth.authenticate(c,creds):result && args(c) && (result==false)
        && InvalidCredentials thrown )
}

7.2.1.4 authorization success.

public class AuthSuccess{
    public pointcut authSuccess(CMSEmployee c):
        call( IAuth.authenticate(c,creds):result && args(c) && (result==true) )
}

7.2.1.5 authentication failure.

public class AuthSuccess{
    public pointcut authSuccess(CMSEmployee c):
        call( IAuth.authenticate(c,creds):result && args(c) && (result==false)
        && NotAuthorizedException thrown)
}

7.2.1.6 witness identity known.

public class WitnessIdentityKnown{
    public pointcut witnessIdentityKnown( ReportId id, String name):
        call( IWitnessReport.enterWitnessIdentity(reportId, name) && args(reportId, reportId) )
}

7.2.1.7 witness identity validated.

public class WitnessIdentityValidated{
    public pointcut witnessIdentityValidated(name):
        call( ICheckIDvalidateIdentity(phonenr, name) && result==true && args(name))
}
7.2.1.8  fake witness.

```java
public class WitnessIdentityValidated{
    public pointcut witnessIdentityValidated(name) :
        call( ICheckID.validateIdentity(phoneno, name) && result==false && args(name) }
```  

7.2.1.9  car crash location known.

```java
public class CarCrashLocationKnown{
    public pointcut carCrashLocationKnown(loc) :
        call( IWitnessReport.enterCarCrashLocation(loc) && args(loc) }
```  

7.2.1.10 car crash confirmed.

```java
public class CarCrashConfirmed{
    public pointcut carCrashConfirmed(location) :
        call( IConfirmCarCrash.confirmCarCrash(location) && result==true && args(name) }
```  

7.2.1.11 fake car crash.

```java
public class CarCrashConfirmed{
    public pointcut carCrashConfirmed(Location loc) :
        call( IConfirmCarCrash.confirmCarCrash(loc) && result==false && args(loc) }
```  

7.2.1.12 strategy selected.

```java
public class StrategySelected{
    public pointcut strategySelected(Strategy strat) :
        execute( IStrategySelected.selectStrategy(strats) && args(result) }
```  

7.2.1.13 employee selection.

```java
public class EmployeeSelection{
    public pointcut employeeSelection(CMSEmployee empl) :
        (call( ICallFwd.forwardCall(Coordinator c, Call call) && args(c)) ||
         (call( IBrief.accept(ResourceProvider rp, Mission m) && args(rp))
```  

7.2.1.14 employee agreement.

```java
public class EmployeeAgreement{
    public pointcut employeeAgreement(CMSEmployee empl) :
        (call( IMissionAccept.missionAccept(ResourceProvider rp, Mission m) && args(rp)) ||
         (call( IWitnessReport.initiateWitnessReport(Coordinator c, Call call) && args(c))
```  

7.2.1.15 no employee available.

```java
public class NoEmployeeAvailable{
    public pointcut noEmployeeAvailable(CMSEmployee empl) :
        (call( IBrief.update(Coordinator c, Mission m) && args(c)) && return==false ||
         (call( IUpdate.update(ResourceProvider rp, Information inf) && args(rp) && return==false)
```
7.2.1.16 action initiation.

```java
public class ActionInitiation{
    public pointcut actionInitiation(CMSEmployee empl) :
        (call( ICallFwd.forwardCall(Coordinator c, Call call) && args(c)) ||
         (call( IAcceptMission.accept(ResourceProvider rp, Mission m) && args(rp))
    }
}
```

7.2.1.17 action performed.

```java
public class ActionPerformed{
    public pointcut actionPerformed(CMSEmployee empl) :
        (call( IWitnessReport.finishWitnessReport(Coordinator c, Mission m) && args(c)) ||
         (call( IUpdate.update(ResourceProvider rp, Information inf) && args(rp) &&
               Information instanceof MissionFinishedNotification )
    }
}
```

7.2.1.18 action failed.

```java
public class ActionFailed{
    public pointcut actionFailed(CMSEmployee empl) :
        call( IUpdate.update(Coordinator c, Information inf) && args(c) &&
              inf instanceof WitnessReportFailedNotification) ||
        call( IUpdate.update(ResourceProvider rp, Information inf) && args(rp) &&
              inf instanceof MissionFailedNotification)
    }
}
```

7.3 Activity 7: Design Aspectual Connectors

In total, we introduced 8 aspectual connectors. To depict them, we have adopted the AO-ADL[1] notation. An aspectual connector binds together a pointcut interface with an advice method.

![Figure 23: Aspectual connector named ‘revise strategy’ for the strategy revision aspect.](image)

![Figure 24: Aspectual connector named ‘witness identity validation’ for the witness identity verification aspect.](image)
Figure 25: Aspectual connector named ‘car crash confirmation’ for the *car crash confirmation* aspect.

Figure 26: Aspectual connector named ‘update car crash’ for the *real-time update* aspect.
Figure 27: Aspectual connector named ‘update log’ for the statistical logging aspect.

Figure 28: Two aspectual connectors named ‘check employee availability’ and ‘set employee availability’ for the resource provider availability aspect.
Figure 29: Aspectual connector named ‘A&A Check’ for the authentication check and authorization check aspects.

7.3.1 Design of Aspectual Connectors

An aspectual connector binds a pointcut interface to advice functionality. Again, we adopt AspectJ syntax to specify the AO connectors.

7.3.1.1 Revise strategy

```java
public aspect ReviseStrategy{
    after(CarCrashId c, Person p, Information i) : updateReceived(c,p,i){
        lookUpCarCrashManagement().recalculateStrategies(c);
    }
}
```

7.3.1.2 Revise strategy

```java
public aspect WitnessIdentityValidation{
    after(ReportId reportId, String name) : witnessIdentityKnown(reportId,name){
        lookUpIdentityValidation().validateIdentity(
            lookUpCarCrashManagement().getReport(reportId).getCall().phoneNb(),Name);
    }
}
```

7.3.1.3 Confirm car crash

```java
public aspect ConfirmCarCrash{
    after(ReportId reportId, Location loc) : carCrashLocationKnown(reportId,loc){
        lookUpCarCrashConfirmation().confirmCarCrash(loc);
    }
}
```

7.3.1.4 Update car crash

```java
public aspect UpdateCarCrash{
    after(CarCrashId c, Person p, Information i) : updateReceived(c,p,i){
        lookUpCarCrashManagement().update(c,i);
    }
}
```

7.3.1.5 Check Employee Availability

Analogous to previous connectors.

**Part II**

**Application of the method: Iteration 2**

In this part of the report, we shortly summarize a second development iteration, in which one additional requirement is adopted in the case study. These results are used further in the discussion regarding effective
pointcut reuse (Section 11).

8 Activity 1: additional requirement: criminal activity monitoring.

For the second iteration, we selected an additional “Safety” requirement from the original requirements documentation [2]: “The system shall monitor criminal activity to ensure safety of rescue resources [...]”. Our particular explicitation (or interpretation) of this requirement is as follows: before dispatching personnel to the crisis site, the system must explicitly consult the nearby police services and request whether criminal activities have been reported that may be related to the car crash, as well as additional information regarding criminal activities that might introduce safety risks when addressing the car crash.

After introducing this new requirement in the use cases and domain models of the CCMS, we applied the method in a second iteration.

First, the domain analysis is extended. For example, the Police Station is introduced as a new type of External Information Provider. Secondly, the use cases are adapted as follows: ‘execute first-aid mission’, ‘execute super observer mission’, and ‘execute witness transportation mission’ are extended to include an explicit check for criminal activities. These steps are illustrated in bold in the ‘execute witness transportation mission’ use case below. Additionally, we introduce the ‘update criminal information’ use case to update the car crash information when criminal activities are reported at the car crash location.

---

**Name:** Execute First Aid Mission *(Implements ‘Execute Mission’)*

**Primary Actor:** Helicopter Pilot

**Main Success Scenario:**

1. The System contacts the Police Station and requests for information regarding criminal activities that might pose a risk to the First-Aid Mission at the car crash location.
2. The Police Station responds that no criminal activities are reported at the car crash location.
3. *(Implements parent UC6, step [...])* The System briefs the Helicopter Pilot with the concrete Mission details: the target Victim to transport, its current Location, as well as the degree of urgency, and the destination Hospital of the Victim, ...
4. *(Implements parent UC6, step [...])* The Helicopter Pilot picks up the Victim and transports him to the Hospital. Finally, she reports this back to the System.

**Alternative Scenarios:**

2b. The Police Station responds that criminal activities have been reported at the crash site. Include: ‘update criminal information’.
   ..b. *(Implement parent UC6, step [...])* The Helicopter Pilot did not manage to execute this mission. Use case fails.

---

In these steps, before dispatching rescue personnel to the crisis location, the system explicitly consults the Police Station to request whether criminal activities have been reported that may be related to the car crash. In case no criminal activities are reported, the use case continues normally. In the alternative scenario, the newly-introduced use case ‘update criminal information’ is executed. This use case is very similar to the ‘update weather information’ use case, in the sense that an external information provider (the Police Station) sends an information update about the car crash. In turn, this might trigger a revision in strategy. Due to space constraints, we do not present this use case in detail.

9 Activities 2-4.

Based on the description above, it is clear that the newly-introduced criminal activity monitoring concern crosscuts the ‘execute first-aid mission’, ‘execute super observer mission’, and ‘execute witness transportation mission’ use cases. In activity 2, this concern is separated from the base use cases. This leads to the
introduction of an aspect use case 'request information regarding criminal activities' that contains the use case steps specific to the criminal activity monitoring aspect.

To compose this aspect use case to the three base use cases, the generalization activity (activity 3) introduces a new extension point named mission initiation in the already-existing abstract use case 'execute mission'.

- **Name:** Request Information regarding Criminal Activities (Implements UC6)
- **Primary Actor:** System
- **Extends:**
  - Execute Mission), before mission initiation.
- **Main Success Scenario:**
  1. The System contacts the Police Station and requests for information regarding criminal activities that might pose a risk to the First-Aid Mission at the car crash location.
  2. The Police Station responds that no criminal activities are reported at the car crash location.
- **Alternative Scenarios:**
  2b. The Police Station responds that criminal activities have been reported at the crash site. Include: update criminal information.

Figure 31 presents a subset of the use case model after applying activity 3. In this diagram, an aspect use case named 'request information regarding criminal activities' was introduced that consists of the use case steps specific to the criminal activity monitoring aspect. The generalization activity (activity 3) has led to the introduction of a new extension point named mission briefing in the already-existing abstract use case 'execute mission'.

10 Activities 5-7.

The adaptations to the architecture are depicted in Figure 30. Again, the color green is used for the newly-introduced elements. This Figure only depicts a subset of the full architecture.

Figure 30: Subset of the CCMS architecture, after introducing the requirements related to the newly-introduced criminal activity monitoring (depicted in green).

The architecture design activity —activity 5— adds a component Police Station Facade that communicates to the external Police Station component to fetch specific information regarding criminal activities.

In activity 6, the mission briefing pointcut interface is added to the Mission Execution component. Finally, in activity 7, an aspectual composition named 'check criminal information' is introduced that composes the mission briefing pointcut interface and the Police Station Facade. This way, before any kind of mission briefing, the system will first consult the police services.
Figure 30 depicts the main adaptations to the architecture. In this diagram, the Police Station Facade component was introduced that handles communication to the external Police Station, more specifically to request information regarding criminal activities related to a car crash. Furthermore, the mission briefing pointcut interface was introduced and implemented by the already-existing Mission Execution component. Finally, an aspectual composition named 'check criminal information' was introduced that composes the mission briefing pointcut interface and the Police Station Facade. This way, before any kind of mission briefing, the system will first consult the police services.

Figure 31: Subset of the full use case diagram after the generalization activity (activity 3). Elements specific to the newly-introduced criminal activity monitoring aspect are depicted in green).

Part III
Discussion

The discussion part in this report is structured as follows: first, we present a discussion with respect to actual pointcut interface reuse. Secondly, we discuss the effort and/or time it took to apply the approach.

11 Types of reuse.

To guide the discussion, we make a distinction between three fundamentally different types of pointcut interface reuse and then we discuss each type of reuse in the context of the CCMS.
11.0.1.6 Type-1: Reuse of existing pointcut interfaces.

This type of reuse occurs when a new aspect is introduced that can be composed with the application in terms of already-existing pointcut interfaces, either fully or partially. Diagram 1 in Figure 32 illustrates this type of reuse graphically.

We can observe this type of reuse for 5 of the 19 pointcut interfaces that were introduced by the method. Table 3 presents these five pointcut interfaces, and lists the aspects that are composed to it.

<table>
<thead>
<tr>
<th>Pointcut Interface</th>
<th>Aspects composed to it</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee selection</td>
<td>statistical logging, resource provider availability</td>
</tr>
<tr>
<td>action failure</td>
<td>statistical logging, resource provider availability</td>
</tr>
<tr>
<td>update received</td>
<td>statistical logging, real-time update, strategy revision</td>
</tr>
<tr>
<td>action initiation</td>
<td>statistical logging, resource provider availability, authentication check, authorization check</td>
</tr>
<tr>
<td>action performed</td>
<td>statistical logging, resource provider availability, authentication check, authorization check</td>
</tr>
</tbody>
</table>

Table 3: The five pointcut interfaces to which more than one aspect is composed.

11.0.1.7 Type-2: Reuse of aspect compositions.

This type of reuse occurs when a new requirement is added that affects an already-existing crosscutting concern. Reuse is successful when the new requirement can be supported by re-implementing an existing pointcut interface, without breaking the pointcut signatures. The re-implementation typically involves expanding the pointcuts to a larger join point set in the base application. This type of reuse is illustrated in diagram 2 of Figure 32.

When introducing the new use case ‘update criminal information’, we already mentioned the similarity to the already-existing ‘update weather information’ use case. Indeed, these use cases have in common that they all update the information regarding the car crash. This commonality was already captured by the abstract use case ‘send car crash information update’ in the first development iteration and thus —as depicted in Figure 31— the ‘update criminal information’ use case was placed in this use case generalization hierarchy.

As a result, the newly-introduced ‘update criminal information’ use case automatically inherits the update received extension point from its parent. In the first iteration, two aspect use cases were specified
to extend this generic use case at this extension point. One of these use cases is the ‘revise strategy’ use case. As a result, the strategy revision aspect automatically extends the ‘update criminal information’ use case: the system will automatically consider a revision in strategy as a result of the information update.

11.0.1.8 Type-3: Reuse within pointcut interface hierarchies.

This type of reuse occurs when the pointcut interfaces themselves are structured hierarchically; i.e. the implementation of high-level pointcut interfaces is done in terms of lower-level ones. This structuring avoids unnecessary duplication of pointcut interface specifications, and whenever the implementation of a lower-level pointcut changes, the higher-level pointcut reuses this change at no extra cost. Figure 32 illustrates this type of reuse in diagram 3.

The introduction of the mission briefing extension point in the abstract ‘execute mission’ use case has lead to the introduction of a pointcut interface hierarchy. This is achieved by employing a technique called pointcut aggregation.

To clarify this, in the first iteration, the action initiation pointcut interface was specified in terms of two specific interface methods:

- action initiation.
  
  ```
  public class ActionInitiation{
    public pointcut actionInitiation(CMSEmployee empl) :
      call( ICallFwd.forwardCall(Coordinator c, Call call) && args(c) )
      || ( call( IAcceptMission.accept(ResourceProvider rp, Mission m)
                  && args(rp) ) )
  }
  ```

In the second iteration however, this pointcut interface is specified to aggregate the newly-introduced mission briefing pointcut interface, as presented below:

- action initiation.
  
  ```
  public class ActionInitiation{
    public pointcut actionInitiation(CMSEmployee empl) :
      call( ICallFwd.forwardCall(Coordinator c, Call call) && args(c) )
      || missionBriefing(empl)
  }
  ```

- mission briefing.
  
  ```
  public class MissionBriefing{
    public pointcut missionBriefing(ResourceProvider rp) :
      call( IAcceptMission.accept(ResourceProvider rp, Mission m)
            && args(rp) )
  }
  ```

The main advantage of this type of multi-level pointcut design is that the intermediary mission briefing pointcut interface implementation can be adapted without affecting the action initiation pointcut interface (given that the pointcut signature itself does not change), and so potential ripple effects are avoided.

12 Time/Effort Spent

Methodology to assess time and effort spent Our assessment of time (and by extension effort) is based on:

- a work log that was kept during the case study
- consulting the calendar that was used to plan the execution of the case study
- E-mail traffic regarding the case study between the authors
- consulting version system history and log messages (subversion).
**Error margin:** Does certainly not exceed two working days. Moreover, even if the absolute values would be off, their relative ratio is still representative.

**Results:** The information from the sources listed above can be summarized as follows:

<table>
<thead>
<tr>
<th>Activities</th>
<th>Period</th>
<th>Calendar Work Days</th>
<th>Percentage of time spent on case study</th>
<th>Actual Working Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity 1</td>
<td>26-05-09 to 30-06-09</td>
<td>29</td>
<td>60%</td>
<td>130.5</td>
</tr>
<tr>
<td>Activities 2 to 4</td>
<td>01-07-09 to 10-07-09</td>
<td>8</td>
<td>70%</td>
<td>42</td>
</tr>
<tr>
<td>Activity 5</td>
<td>22-07-09 to 04-08-09</td>
<td>15</td>
<td>80%</td>
<td>91.2</td>
</tr>
<tr>
<td>Activities 6 to 7</td>
<td>04-08-09 to 10-08-09</td>
<td>4</td>
<td>80%</td>
<td>22.8</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>26-05-09 to 10-08-09</strong></td>
<td><strong>n/a</strong></td>
<td><strong>n/a</strong></td>
<td><strong>286.5</strong></td>
</tr>
</tbody>
</table>

In total 286.5 person hours were spent on the case study. Figure 33 presents a break-down of these man hours per activity. This figure indicates how much of this effort can be accounted to specifically applying the method (activities 2 to 4, and 6 to 7): in total, 64.8 man hours were spent on activities specific to the method, which is around 23% of the total. This amount of effort is consistent with the amount of effort that was needed for applying our method in previous case studies [4].

![Figure 33: Breakdown of the man hours spent on the case study per activity.](image)

**References**


