Extending the User Requirements Notation with Aspect-oriented Concepts

Gunter Mussbacher and Daniel Amyot

SITE, University of Ottawa, 800 King Edward, Ottawa, ON, K1N 6N5, Canada
{gunterm, damyot}@site.uottawa.ca

Abstract. In November 2008, the User Requirements Notation (URN) was approved as Recommendation Z.151 by the Standardization Sector of the International Telecommunication Union (ITU-T). URN is the first and currently only standard that supports both goal-oriented and scenario-based modeling for requirements engineering activities. The Aspect-oriented URN (AoURN) is a recent extension of URN that combines goals, scenarios, and aspects in one framework. AoURN is a candidate for future versions of the URN standard. We first summarize the basic concepts and notation of AoURN and then discuss advanced features of AoURN that are necessary for large-scale modeling. Based on our experience with AoURN modeling, we present a list of requirements including rationale for an aspect-oriented extension of Z.151 that succinctly expresses the required features to evolve URN into a complete aspect-oriented modeling framework.

Key words: User Requirements Notation, Goal-oriented Requirement Language, Use Case Maps, Aspects, Aspect-oriented Modeling, Aspect-oriented Requirements Engineering, Aspect-oriented User Requirements Notation

1 Introduction

The recent Recommendation Z.151 (11/08) defines the User Requirements Notation (URN) [9], a modeling language for requirements engineering and high-level design that incorporates goal-oriented and scenario-based models in one framework. While scenario models have been regarded as an essential tool for software development for a long time, goal models are a more recent development [5, 22, 26, 28]. In URN, goal models created with the Goal-oriented Requirement Language (GRL) are complemented with scenario models created with Use Case Maps (UCMs). Even though Recommendation Z.151 is now available, new developments in software engineering must be considered in the context of evolving URN. One of the most exciting developments in software engineering over the last decade is the emergence of aspect-oriented modeling (AOM) [4], promising better encapsulation of crosscutting concerns at the requirements and architecture stage which in turn may lead to greater maintainability, reusability, and scalability of requirements models. Crosscutting concerns in the context of requirements engineering are for example features, scenarios, and non-functional

requirements (NFRs) that cannot be properly encapsulated with only traditional requirements engineering techniques. Many aspect-oriented approaches for such techniques have been proposed (e.g., for use cases [2, 3, 10], viewpoints [24], problem frames [12], and UML models [6, 27]).

For the last three years, the Aspect-oriented User Requirements Notation (AoURN) has been developed [13–21, 23]. During this time period, the initial basic concepts of AoURN have been augmented with several advanced features. We summarize all of these advanced features together here in one publication for the first time, and furthermore discuss the motivation of these advanced features. Based on our experiences in combining URN and aspect-oriented concepts, numerous case studies, and feedback from industrial collaborators, we are now in a position to formulate requirements for the extension of URN with aspects. This approach follows the example of URN itself, as requirements for URN were first published in Recommendation Z.150 (02/03) "User Requirements Notation (URN) – Language requirements and framework" [8] and later Z.151 was defined based on these requirements. The goal therefore is to amend Z.150 with requirements for aspect-oriented modeling based on the requirements presented here. Furthermore, Z.151 itself contains additional requirements that describe in detail the semantics of the dynamic behavior of UCMs when interpreted (i.e., traversed) by a path traversal mechanism. Hence, the second set of requirements presented here is formulated for the UCM path traversal to enable the traversal of aspect-oriented UCM models.

In the remainder of this paper, Section 2 gives a brief overview of Recommendation Z.151, the User Requirements Notation (URN). Section 3 explains the basic concepts of the Aspect-oriented User Requirements Notation (AoURN). Section 4 then addresses advanced features of AoURN required for large-scale modeling. A simple web-based application serves as an example to illustrate the basic and motivate the advanced features of AoURN. Section 5 presents the list of requirements for extending URN with aspect-oriented concepts, grouped into general requirements and requirements for the path traversal of Use Case Maps (UCM). Finally, Section 6 concludes the paper and identifies future work.

2 Overview of Z.151 – User Requirements Notation

The User Requirements Notation (URN) [1, 9] supports the elicitation, analysis, specification, and validation of requirements. URN captures early requirements in a modeling framework containing two complementary sub-languages called Goal-oriented Requirement Language (GRL – for goal-oriented modeling) and Use Case Maps (UCMs – for scenario-based modeling). GRL models are used to describe and reason about non-functional requirements (NFRs), quality attributes, and the intentions of system stakeholders, whereas UCM models are used for operational requirements, functional requirements, and performance and architectural reasoning. While GRL identifies at a very high level of abstraction possible solutions to be considered for the proposed system, UCM models describe these solutions in more detail. In summary, URN has concepts for the

specification of stakeholders, goals, non-functional requirements, rationales, behaviour, actors, scenarios, and structuring.

A GRL model consists of *intentional elements* (e.g., softgoals (⬭), hard goals (⬭), and tasks (⬭)) connected together with different types of links. Intentional elements may be assigned to stakeholders called *actors* (⬭). *Contribution links* (⟶) indicate positive (+) or negative (–) impact of intentional elements on each other. *Correlation links* (⇢) are similar to contribution links in that they also indicate impact but are used to describe side effects rather than desired impacts. *Decomposition links* (+—) allow the decomposition of intentional elements into sub-elements. AND, (inclusive) OR, and XOR decompositions are supported.

For example in Fig. 1, two stakeholders, the Customer and the DVD Store, are shown. The customer's goal graph has three intentional elements with positive contribution links between them, whereas the store's goal graph is a little more complicated and considers two alternatives, Online Store and Traditional Store, modeled as tasks.

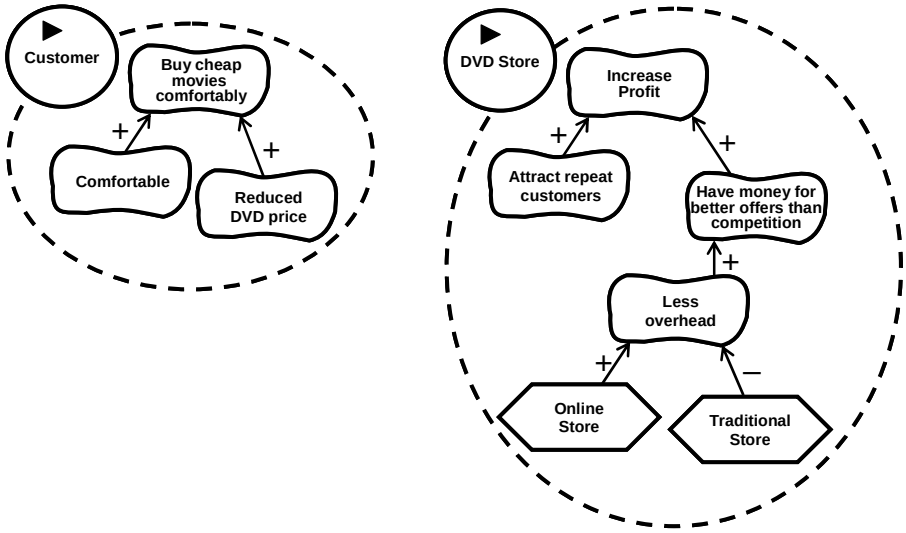

Fig. 1. The Customer and DVD Store Stakeholders of a Simple Online DVD Store System

A UCM model consists of a path that begins at a *start point* (●, e.g., buy) and ends with an *end point* (▮, e.g., bought). A path may contain *responsibilities* (✕, e.g., processOrder), identifying the steps in a scenario, and notational symbols for alternative (⌒) and concurrent (⊣⊢) branches. Path elements may be assigned to a *component* (▭, e.g., DVD Store). *Stubs* (◇) are containers for sub-models called *plug-in maps*. Drilling into a stub leads to a submap that provides more

details, thus allowing for hierarchical structuring of UCM models. A binding between the stub and elements on the plug-in map precisely defines how the scenario continues from the parent map to the submap and back to the parent map.

Furthermore, *URN links* (▶) are used but are not limited to establish traceability by relating tasks (i.e., possible solutions) or actors in goal models to their representation in the UCM model (e.g., maps, responsibilities, and stubs or components that further describe the linked GRL elements). For example, the GRL Customer stakeholder in Fig. 1 is linked to the UCM Customer component in Fig. 2. Finally, URN allows the definition of *metadata* (<<MetadataName>>) for any URN modeling element, thus providing an extension mechanism that permits the definition of profiles for URN.

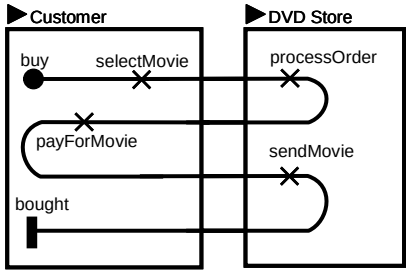

Fig. 2. The Buy Movie Use Case of a Simple Online DVD Store System

The most comprehensive URN tool available to date is the Eclipse plug-in jUCMNav [11]. jUCMNav is a full editor for GRL and UCM models that ensures that only syntactically correct URN models are created. The tool manages hierarchical UCM models consisting of several layers of maps and plug-in maps and allows the requirements engineer to navigate easily through them. In addition, the tool provides standard analysis features for URN models, i.e., global quantitative and qualitative tradeoff analysis for conflicting stakeholder goal models as well as validation and (regression) testing of UCM models including the detection of undesired feature interactions. jUCMNav supports the built-in traceability and profiling features of URN as well as several transformations. Structured, textual use cases may be imported into URN models and URN models may be transformed into more detailed scenario languages such as message sequence charts but also into performance models. Furthermore, jUCMNav synchronizes URN models with requirements management tools and therefore allows URN models to be managed together with other types of requirements. OCL constraints on URN models can also be defined and verified by jUCMNav. Some support for aspect-oriented modeling is already available for jUCMNav. Further AO functionality is being prototyped and will be added to the tool in the near future. Finally, the tool also supports more advanced research on URN-

based business process monitoring and runtime adaptation. For more details about URN, visit the URN Virtual Library [25].

3 Basic Concepts of AoURN

The Aspect-oriented User Requirements Notation (AoURN) [15–19, 21, 23] extends the User Requirements Notation (URN) with aspect-oriented concepts, allowing modelers to better encapsulate crosscutting concerns which are hard or impossible to encapsulate with URN models alone. AoURN adds aspect concepts to URN's sub-languages, leading to and integrating Aspect-oriented GRL (AoGRL) [17, 23] and Aspect-oriented UCMs (AoUCM) [16, 18, 19]. The three major aspect-oriented concepts that have to be added to URN are concerns, composition rules, and pointcut expressions. Note that the term aspect refers to a crosscutting concern, while the term concern encompasses both crosscutting and non-crosscutting concerns. These are core concepts of many aspect-oriented modeling (AOM) techniques. In terms of aspect-oriented programming (AOP, e.g. with AspectJ [7]), the concept of a crosscutting concern in AOM relates to the concept of an aspect and the aspect's advice in AOP, the concept of composition rules in AOM encompasses the common before/after/around operators in AOP, and the concept of pointcut expressions is the same in AOM and AOP.

A *concern* is a new unit of encapsulation that captures everything related to a particular idea, feature, quality, etc. AoURN treats concerns as first-class modeling elements, regardless of whether they are crosscutting or not. Typical concerns in the context of URN are stakeholders' intentions, NFRs, and use cases. AoURN groups all relevant properties of a concern such as goals, behavior, and structure, as well as pointcut expressions needed to apply new goal and scenario elements to a URN model or to modify existing elements in the URN model.

Pointcut expressions are patterns that are specified by an aspect and matched in the URN model (often referred to as the base model). If a match is found, the aspect is applied at the matched location in the base model. The *composition rule* defines how an aspect transforms the matched location. AoURN uses standard URN diagrams to describe pointcut expressions and composition rules (i.e., AoURN is only limited by the expressive power of URN itself as opposed to a particular composition language). AoURN's aspect composition technique can fully transform URN models.

Section 2 has already introduced a URN model with three concerns for the example application: the Customer stakeholder, the DVD Store stakeholder, and the Buy Movie use case. These concerns will now be expanded and new concerns will be added in the remainder of this section with the help of basic AoURN features.

AoURN adds the ability to define pointcut expressions and then compose aspects with the URN model. GRL pointcut expressions are shown on a *pointcut graph* and make use of *pointcut (deletion) markers* (✧,✕) to indicate the pattern to be matched. All elements without pointcut markers are added to the matched location in the GRL base model, while elements with a pointcut deletion marker

are removed. The composition rule is therefore defined by the set of links between elements without pointcut markers and elements with pointcut markers. Generic, reusable goals and tasks of an aspect may be described in more detail in separate goal graphs called *aspect graphs*.

For example, Fig. 3 depicts four pointcut graphs showing how the DVD Store and the Customer stakeholders are connected (Fig. 3.a), how the Buy Movie use case and the new Earn/Redeem Movie Points use case impact the goal model (Fig. 3.b and Fig. 3.c), and how the new Security NFR concern impacts the use cases (Fig. 3.d). Note that the tasks defined on the pointcut graphs of the use case concerns have URN links to UCM elements (e.g., Process online and send by mail is linked to processOrder and sendMovie shown in Fig. 2). Furthermore, a reusable aspect graph of the Security NFR concern is shown in Fig. 4, explaining how security can be generally achieved by modeling knowledge from the security domain.

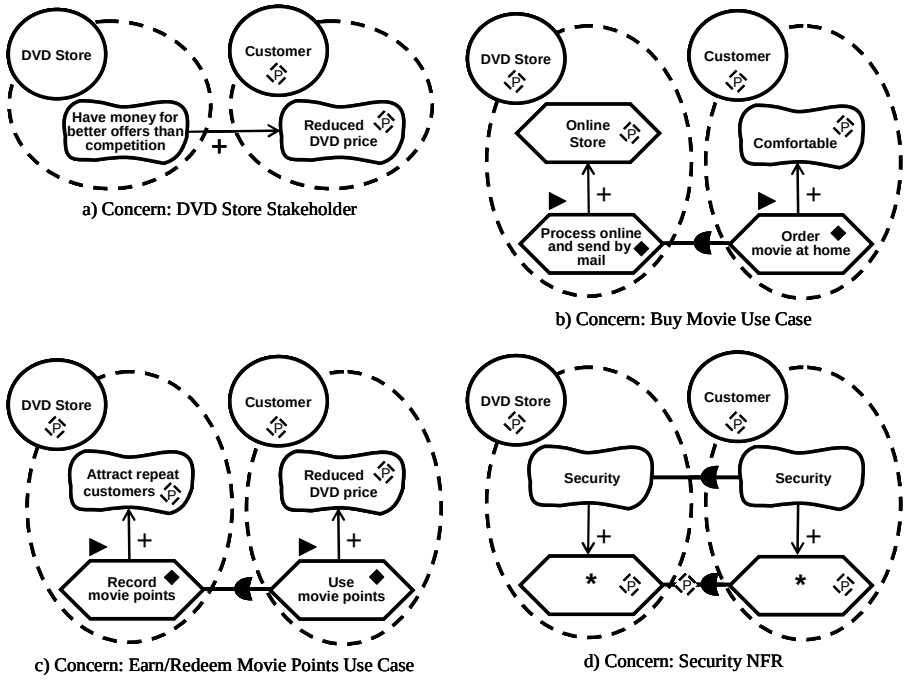

Fig. 3. AoGRL's Pointcut Graphs for a Simple Online DVD Store

The pointcut expressions in Fig. 3.a, Fig. 3.b, and Fig. 3.c are rather straightforward and the composition of these aspects with the base model is simple. The base locations affected by these three aspects are indicated by four *aspect markers* (◆, e.g., Reduced DVD price) in Fig. 5. The pointcut expression of the Security NFR concern in Fig. 3.d connects the reusable aspect graph of the Security NFR

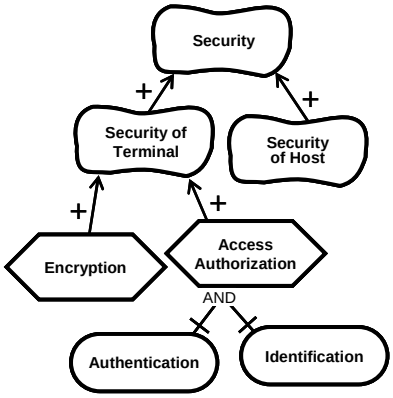

Fig. 4. AoGRL's Aspect Graph for the Security NFR Concern

concern with the application-specific goal model of the online DVD store. The pointcut expression matches any two tasks with a dependency link as long as one task resides within the DVD Store stakeholder while the other task resides within the Customer stakeholder. The base locations affected by the Security NFR concern are again indicated by aspect markers (i.e., the four tasks in Fig. 3.b and Fig. 3.c).

When an aspect marker is selected, the modeler is taken to the *AoView* of the aspect with only those aspectual properties of the pointcut graph highlighted that are relevant to the aspect marker. The aspect markers and the AoViews allow the requirements engineer to reason about the composed model. The concept of composed models in AOM thus relates to aspects being woven into the base in AOP. Essentially, aspect markers in AoURN are similar to the advice markers in AspectJ shown on the left-hand side of the Java Eclipse editor [7].

The AoViews of the aspect markers in Fig. 5 are simply the pointcut graphs from Fig. 3 without the symbols for the pointcut markers – i.e., the AoView of the aspect marker for Comfortable is Fig. 3.b, for Reduced DVD price it is Fig. 3.a and Fig. 3.c because Reduced DVD price is matched by two pointcut graphs, for Attract repeat customers it is Fig. 3.c (see Fig. 6.a), and for Online Store it is Fig. 3.b. The two examples illustrated in Fig. 6.b and Fig. 6.c show the AoViews corresponding to the aspect markers of the Security NFR. The AoView of the aspect markers of Order movie at home and Process online and send by mail is shown in Fig. 6.b, while the AoView of the aspect markers of Record movie points and Use movie points is shown in Fig. 6.c. Note how parameterized elements of the pointcut expression are replaced by their actual matches in the AoViews in Fig. 6.b and Fig. 6.c. By using the models defined by the requirements engineer for the AoViews, it is possible to view the composed AoURN model without having to resolve complex layout issues (this applies to AoGRL and AoUCM).

Similarly to GRL pointcut expressions, UCM pointcut expressions define the pattern to be matched with a *pointcut map*. Grey start and end points on

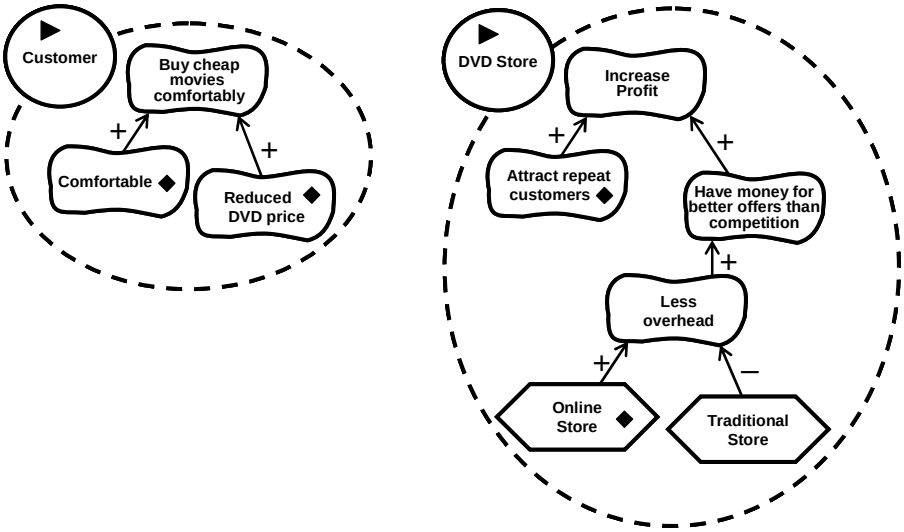

Fig. 5. AoGRL's Aspect Markers for a Simple Online DVD Store System

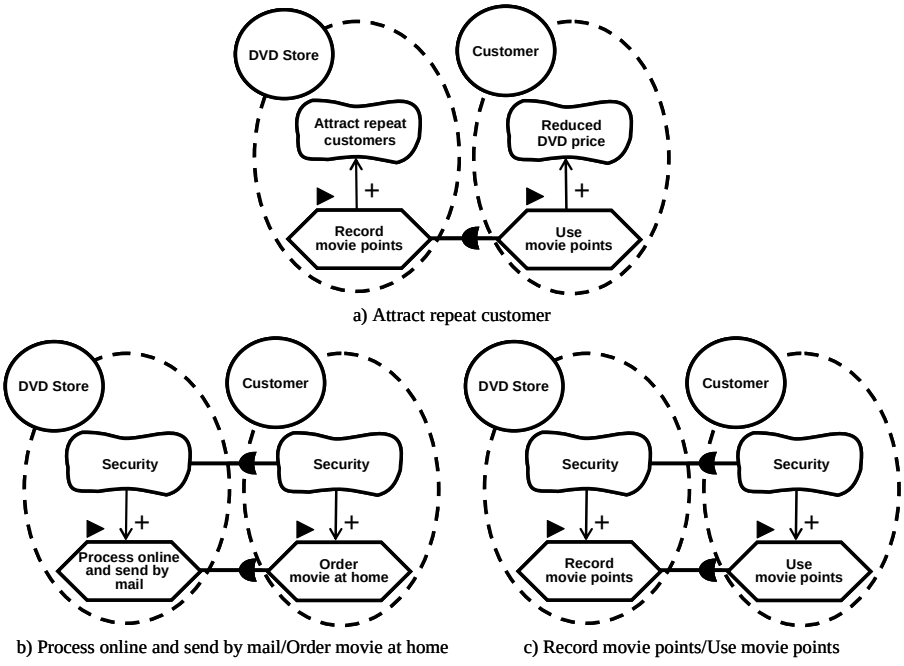

Fig. 6. AoGRL's AoView for five Aspect Markers

the pointcut map are not part of the pointcut expression but rather denote its beginning and end. The aspectual properties are shown on a separate *aspect map*, allowing the pointcut expression and the aspectual properties to be individually reused. The aspect map is linked to the pointcut expression with the help of a *pointcut stub* (). The causal relationship of the pointcut stub and the aspectual properties visually defines the composition rule for the aspect, indicating how the aspect is inserted in the base model (e.g., before, after, optionally, in parallel or anything else that can be expressed with the UCM notation). The *replacement pointcut stub* () is a special kind of pointcut stub, indicating that the aspect is replacing the matched base elements.

For example, the purpose of the new Communication concern in Fig. 7 is to define in more detail the interaction between the customer and the online DVD store when the customer selects a movie. The pointcut map therefore matches against the selectMovie and processOrder responsibilities in the Customer and DVD Store components, respectively. The bindings of the pointcut stub connect the in-path of the stub with the start point of the pointcut map and the out-path of the stub with the end point of the pointcut map. As the pointcut stub on the aspect map is a replacement pointcut stub, the matched responsibilities are replaced with the aspectual properties described on the aspect map. The aspect map defines that selectMovie and processOrder are reinserted and explicit request and reply responsibilities as well as a waiting place are added, specifying that the customer has to wait for the response of the online DVD store. Note that the specification of this aspect is rather problematic as will be discussed in Section 4. However, it serves its purpose here to introduce the main concepts of AoUCM.

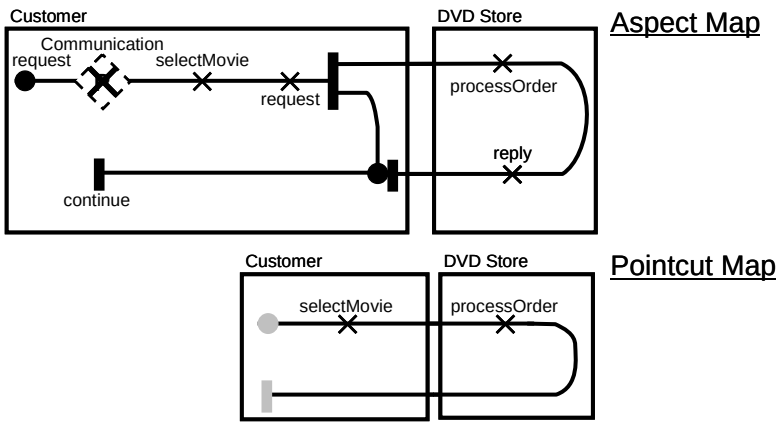

Fig. 7. AoUCM's Aspect Map and Pointcut Map for the Communication Concern

Similar to GRL aspect markers, UCM aspect markers (◆) also indicate affected base locations in the UCM model. If the aspect adds elements before or after the base location matched by the pointcut expression, the aspect marker is added before or after the base location, respectively. In the case of a replacement, two aspect markers are added, one before and one after the replaced base elements. Contrary to AoGRL, a UCM aspect marker is not just an annotation but is rather a kind of stub that links the base model with a submap, i.e., the aspect map. Bindings between the aspect marker and the aspect map are created automatically by AoURN's composition mechanism. Figure 8 shows the aspect markers added to the use case from Fig. 2 because the pointcut expression in Fig. 7 matches selectMovie and processOrder in this use case.

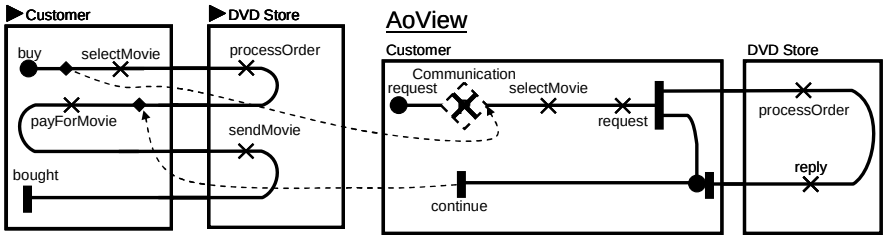

Fig. 8. AoUCM's Aspect Markers and their corresponding AoView

Fig. 8 also shows the AoView for the aspect markers, highlighting the portion of the Communication aspect map that is inserted. When the first aspect marker is reached during the Buy Movie scenario, the scenario continues with the aspectual behavior on the aspect map (right after the pointcut stub). When the aspect map's end point is reached, the scenario continues with the second aspect marker, thus skipping the replaced base elements. If the aspect does not replace base elements but e.g. simply adds elements before or after the matched base location, then the scenario returns from the aspect map to the same aspect marker (i.e., the aspect marker has bindings to and from the aspect map). Note that the dashed arrows depicting bindings are only added for illustration purposes and are not part of AoUCM's concrete syntax. The jUCMNav tool manages bindings and hierarchical UCM models much more effectively.

Finally, aspects may depend on or conflict with each other. AoURN models dependencies and conflicts among concerns and the resolution thereof with the help of the *Concern Interaction Graph* (CIG) [15]. Many aspect interactions can be resolved with precedence rules. The CIG is a specialized GRL goal graph that uses dependencies, correlations, and intentional elements to model such precedence rules. The precedence rules then govern the order in which concerns are applied to a URN model.

4 Advanced Features of AoURN

The pointcut expression in Fig. 7 is problematic because only a very specific interaction between the customer and the online DVD store is matched. It, however, is very likely that many if not all interactions between the customer and the online DVD store use the request/reply pattern defined by the aspect in Fig. 7. To rectify this problem, the pointcut expression can easily be adapted to match against any two responsibilities (simply by using the wildcard * instead of concrete names for the responsibilities).

This is better but still very fragile with respect to rather small changes to the base model. If another responsibility is added in the base model after processOrder (e.g., rewardReferrer), then the pointcut expression will no longer match. The pointcut expression expects only two responsibilities before the path crosses back into the Customer component. Therefore, the pointcut expression must be made more flexible. This is achieved by the *anything pointcut element* (.....) as shown in Fig. 9. The anything pointcut element matches against an arbitrary sequence of base elements.

At this point, the aspect itself is not generic enough and therefore cannot be reused easily, because it also specifies very concrete elements of the base model (i.e., Customer, DVD Store, selectMovie, processOrder). In order to model a reusable aspect, it must be possible to reuse base elements matched by the pointcut expression. This is achieved with *variables* denoted by the prefix $ as shown in Fig. 9 on the aspect map and pointcut map.

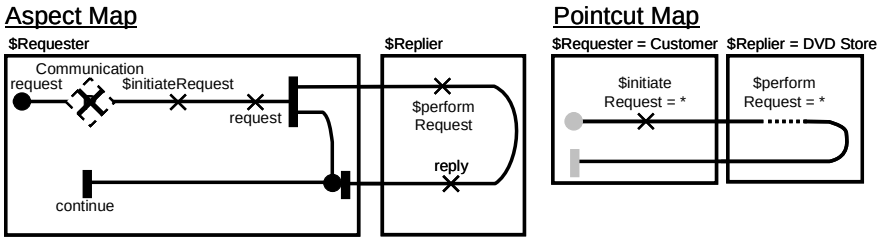

Fig. 9. Reusing Matched Elements with Variables and the Anything Pointcut Element

The pointcut expression in Fig. 9 now matches against selectMovie, processOrder, and rewardReferrer as well as payForMovie and sendMovie (hence there are two pairs of aspect markers in Fig. 10). Furthermore, when the aspect is applied to a location, the variables in the aspect are replaced by the matched elements (i.e., $initiateRequest is replaced by selectMovie or by payForMovie; $performRequest is replaced by processOrder and rewardReferrer or by sendMovie; $Requester is replaced by Customer; and $Replier is replaced by DVD Store as shown in the AoViews in Fig. 10).

Note that the anything pointcut element is only available for AoUCM. It is not available for AoGRL because the highly interconnected nature of goal

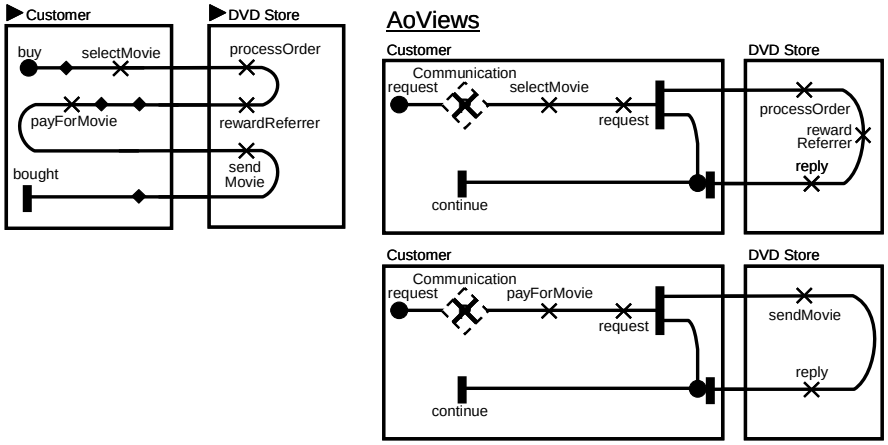

Fig. 10. AoUCM's Aspect Markers and their corresponding AoView

graphs leads to an undesired explosion of matches. Furthermore, AoGRL does not require variables because GRL pointcut expressions are not separated from the description of aspectual properties. Although this is an advantage, the disadvantage is that individual reuse of pointcut expressions and aspectual behavior, while possible with AoUCM, is not possible with AoGRL.

The pointcut expression in Fig. 9 now captures all interactions between the customer and the DVD store. However, some interactions may have to be treated differently, e.g., asynchronous communication may be more appropriate for some interactions. This may lead to pointcuts with complicated regular expressions. Annotation-based matching is a common approach for aspect-oriented programming languages and can be effectively used to address this issue. AoURN supports *annotation-based matching* with metadata for both AoGRL and AoUCM models. Metadata may be specified in the pointcut expression and the base model may also be annotated with the metadata, explicitly identifying the locations to which the aspect should be applied. The metadata is then simply taken into account by the matching algorithm of AoURN.

The aspect map in Fig. 9, which describes a communication mechanism, may be further improved by adding checks for corrupted replies and a way to retry the interaction. Figure 11 shows the updated aspect map. This, however, leads to an ambiguous situation when the aspect map is connected to the aspect marker through bindings. It is not clear which of the two start points should be connected and which of the two end points. This is addressed by the definition of *local start and end points* (and , e.g., retry and fail in Fig. 11), which by definition are never connected by bindings with aspect markers.

An attempt to describe in more detail the Earn/Redeem Movie Points use case introduced in Fig. 3 uncovers another problem. The Buy Movie (BM) use case and the Earn/Redeem Movie Points (ERMP) use case are heavily inter-

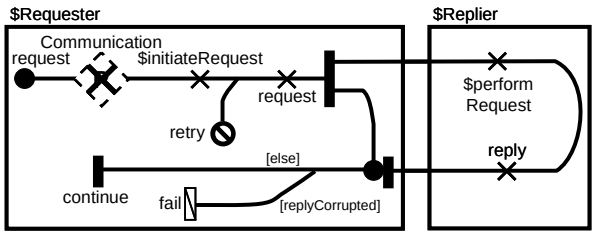

Fig. 11. Local Start and End Points for Aspects

twined with each other. First a membership form needs to be filled (ERMP), then the order is processed (BM), then movie points may be redeemed if enough points are available (ERMP). If there are not enough points available (ERMP), then the customer pays for the movie as usual (BM) but earns movie points (ERMP) once the movie has been sent (BM). The problem with intertwined use cases is that either both use cases have to be modeled together (i.e., different concerns are tangled with each other) or many small aspects add the individual steps of one use case to the other, making it hard to understand how these small aspects relate to each other. Intertwined use cases are a very common phenomenon. AoUCM uses *interleaved composition* to address this problem by allowing multiple pointcut stubs to be defined for a single aspect map.

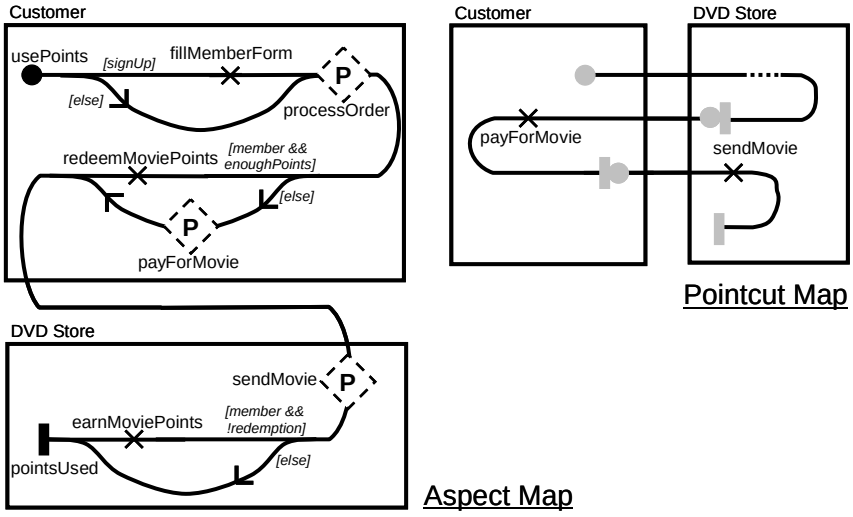

Fig. 12. Interleaved Composition

A requirements engineer can understand the ERMP use case simply by looking at its definition, i.e., the aspect map in Fig. 12. First, a member form must be filled if the customer is not yet signed up, then movie points may be redeemed if enough points are available, and finally movie points may be earned if the member did not redeem movie points in the same transaction. In addition, the pointcut stubs identify how the BM use case is interleaved with the ERMP use case. The member form must be filled before processing the order, redeeming of movie points may occur instead of paying for a movie, and earning movie points happens after the movie was sent.

As shown in the pointcut expression in Fig. 12, interleaved composition is achieved with a series of UCM paths, established by connecting together grey end and start points. Pairs of grey end and start points are ignored by the matching algorithm and the pointcut expression is therefore easily matched against the Buy Movie use case described in Fig. 10. The additional grey end and start points, however, are used when connecting a series of pointcut stubs to a series of UCM paths on the pointcut map. All examples until now featured only pointcut maps with one grey start and one grey end point and aspect maps with only one pointcut stub. The binding to the pointcut stub on the aspect map was therefore straightforward – the in-path of the pointcut stub is connected to the grey start point and the out-path to the grey end point. In Fig. 12, the processOrder pointcut stub is connected to the first path segment of the pointcut expression, the payForMovie pointcut stub to the second path segment, and the sendMovie pointcut stub to the third and last path segment. Note how the causal relationship of the individual steps of the Earn/Redeem Movie Points use case are clearly described on the aspect map and how the bindings of pointcut stubs to UCM paths define the interleaving of the two use cases, while the pointcut expression remains fairly simple. Interleaved composition is powerful yet it is seldom supported in other AOM and AOP approaches.

Fig. 13 highlights two scenarios of the Earn/Redeem Movie Points use case composed with the Buy Movie use case. The first is the initial scenario where a new member signs up to the movie points program, while the second scenario shows a redemption of movie points. The dashed arrows have been added to illustrate the bindings created automatically by the composition mechanism of AoURN, allowing the reader to follow along the scenario as it unfolds. Note that the jUCMNav tool does not require the bindings to be visualized but allows the modeler to conveniently navigate from stubs to plug-in maps or aspect markers to aspect maps by clicking on the stub or aspect marker, respectively. This results in a much less confusing experience for the requirements engineer than Fig. 13 seems to convey.

The operationalization of the Security NFR concern from Fig. 3.d can also be modeled concisely with interleaved composition. The pointcut expression matches against the request responsibility, the AND-fork, $performRequest, reply, and the path crossing back into the $Requester component in the Communications concern. Encryption processing is then added after the request and reply

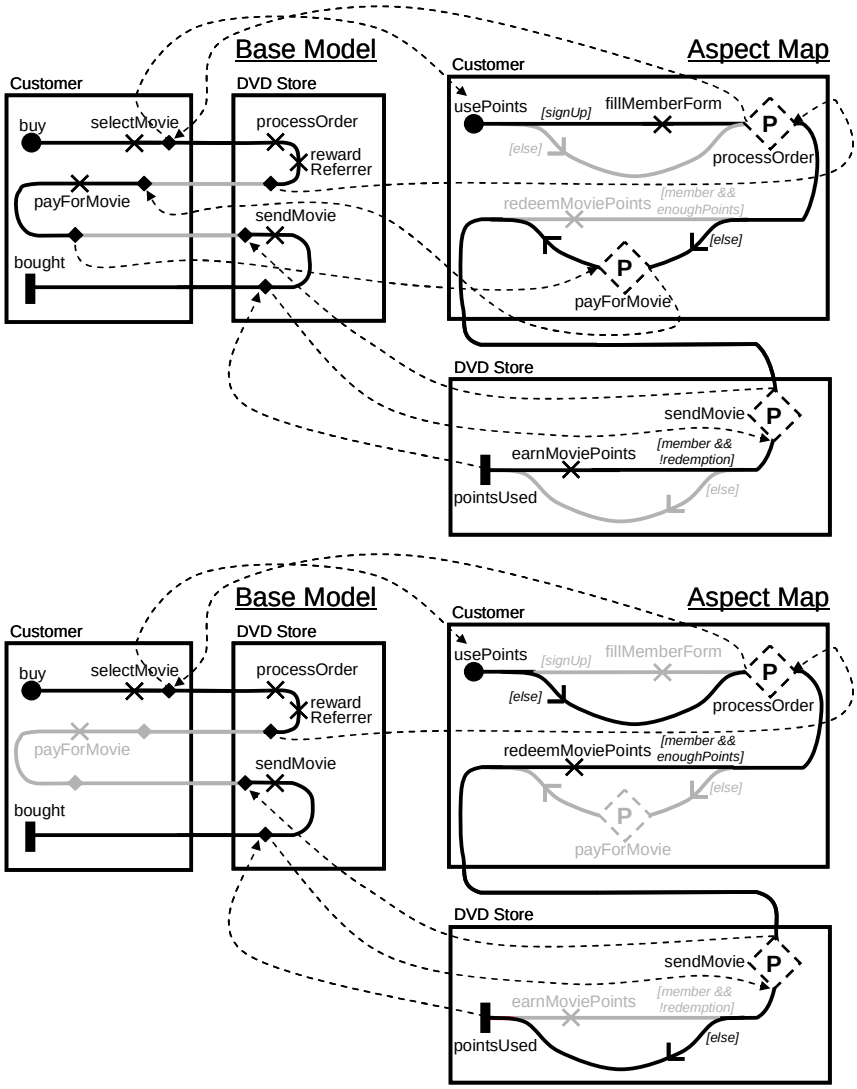

Fig. 13. Two Highlighted Scenarios of the Earn/Redeem Movie Points Use Case

responsibilities, while decryption processing is added before $performRequest and after crossing back into the $Requester component.

Some of the aspect markers in Fig. 13 only have a binding to the aspect map, some only one from the aspect map, and some both. The concrete syntax for all three cases, however, is the same. This makes it difficult for the requirements engineer to understand the impact of an aspect simply by looking at the aspect marker. AoUCM differentiates between these three cases (◆ ... standard = to and from bindings, ◆▸ ... beginning of a replacement = only to bindings, ▸◆ ... end of a replacement = only from bindings), making it possible to *identify replacements immediately.*

5 Requirements for Extending Z.151 with Aspects

This section proposes requirements for improving modularity and separation of concerns in URN by extending it with aspect-oriented concepts (Table 1). Some of these requirements may not require any changes to the current version of URN, especially if a proposed solution for AOM with URN uses existing modeling constructs specialized by metadata in a URN profile. These requirements are still included here as this set of requirements aims to be complete with respect to coverage of key aspect-oriented concepts.

6 Conclusion

We have presented the Aspect-oriented User Requirements Notation (AoURN), an extension to the User Requirements Notation (URN). Basic and advanced features of AoURN were illustrated and discussed. Based on these features as well as our experience with AoURN modeling in general, we proposed a list of requirements that can be used to evolve URN into a complete aspect-oriented modeling environment for requirements engineering activities. In future work, the requirements for supporting the analysis and validation features of URN in an aspect-oriented way as well as the requirements for the path traversal mechanism will still have to be refined. Furthermore, recent work on analyzing semantic aspect interactions with the help of GRL graphs could be considered for inclusion into an updated version of the URN standard. Finally, the notions of aspect markers and pointcut expressions spanning model types with the help of URN links could be applied to other AOM techniques to address layout issues of composed models and support multi-model aspects.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada, through its programs of Discovery Grants and Postgraduate Scholarships.

Requirement (URN shall allow…)	Rationale
GRL and UCM model diagrams and elements to be grouped into concerns.	Basic premise of aspect-oriented modeling.
Patterns (i.e., pointcut expressions) to be defined for a concern.	Basic premise of aspect-oriented modeling.
Patterns to be defined that are at the same level of complexity as URN itself.	A pattern language (i.e., pointcut language) that is not able to capture all constructs of URN is limiting the expressiveness and flexibility available to the requirements engineer.
Parameterized patterns to be defined.	Parameterized patterns allow concerns to transform a large numbers of base locations.
Patterns that include logical operators AND, OR, or NOT.	Required to fine-tune pattern descriptions.
Patterns that include metadata.	Required to fine-tune pattern descriptions.
Patterns that match against an arbitrary sequence of UCM base elements.	Required to safe-guard successful pattern matches against small changes in the base model or to efficiently capture variations in the target pattern.
The reuse of matched UCM elements in the description of the concern properties.	Required to specify highly reusable concerns.
Patterns to be defined over both model types.	Aspect-oriented extensions to URN must take advantage of the URN framework.
Purely syntactic model elements to be ignored when matching the pattern against the base model.	Required to safe-guard successful pattern matches against inconsequential changes in the base model.
Semantically equivalent models to be matched even if their syntactic representation is different.	Required to safe-guard successful pattern matches against inconsequential changes in the base model.
Composition rules to be defined for a concern.	Basic premise of aspect-oriented modeling.
Composition rules to be defined that are at the same level of complexity as URN itself.	Composition rules that are not able to express all constructs of URN are limiting the expressiveness and flexibility available to the requirements engineer.
Interleaved composition to be defined.	Scenarios are often intertwined with each other. Interleaved composition allows scenarios to be defined separately from each other without losing context information and increasing pattern complexity.
Locations to be transformed by a concern only if the transformation is indicated at the locations.	Basic feature of aspect-oriented modeling and programming.
Locations to be transformed by a replacement transformation of a concern only if the replacement transformation is indicated at the locations in a different way than for locations that are transformed in some other way.	It is important for the requirements engineer to understand at a glance the impact of concerns on the matched base locations.
Concern dependencies and conflicts as well as their resolutions to be modeled.	Concerns interact with each other. Without support for the resolution of these interactions, the model specification is ambiguous and may result in undesired interactions.
The individual, separate reuse of concern properties and patterns (patterns may only be reused for UCM but not for GRL).	A goal of aspect-oriented modeling is to provide highly reusable assets. URN must support this by allowing concern properties and patterns to be reused separately from each other.
The base model composed with all concerns to be viewed without requiring the user to resolve complex layout issues.	Requirements engineers need to have access to a fully composed model for further analysis. URN must support this.
Its analysis and validation features to be performed in an aspect-oriented way.	This is a very general requirement at this point. However, any aspect-oriented extension to URN must not prevent or limit the current analysis and validation features.
The UCM Path Traversal to traverse the specified concern behavior upon arrival at a location transformed by a concern before continuing with the traversal at a base location.	This is also quite general at this point and will have to be refined given a particular aspect-oriented extension of URN.

Table 1. Requirements for Extending Z.151 with Aspect-oriented Concepts

References

1. Amyot, D. and Mussbacher, G.: Development of Telecommunications Standards and Services with the User Requirements Notation. *Workshop on ITU System Design Languages 2008*, Geneva, Switzerland (September 15-16, 2008)
2. Araújo, J. and Coutinho, P.: Identifying Aspectual Use Cases Using a Viewpoint-Oriented Requirements Method, *Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop of the 2nd International Conference on Aspect-Oriented Software Development*, Boston, USA (17 March 2003)
3. Araújo, J. and Moreira, A.: An Aspectual Use Case Driven Approach. *VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2003)*, Alicante, Spain (November 2003)
4. Chitchyan, R. et al.: *Survey of Analysis and Design Approaches*. AOSD-Europe Report ULANC-9 (May 2005). http://www.aosd-europe.net/deliverables/d11.pdf (accessed April 2009)
5. Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J.: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, Dordrecht, USA (2000)
6. Clarke, S. and Baniassad, E.: *Aspect-Oriented Analysis and Design: The Theme Approach*. Addison Wesley (2005)
7. Eclipse.org: *The AspectJ Project*; http://www.eclipse.org/aspectj/ (accessed June 2009)
8. ITU-T – International Telecommunication Union: *Recommendation Z.150 (02/03), User Requirements Notation (URN) – Language Requirements and Framework*, Geneva, Switzerland (2003)
9. ITU-T – International Telecommunication Union: *Recommendation Z.151 (11/08): User Requirements Notation (URN) – Language definition*, Geneva, Switzerland (2008)
10. Jacobson, I. and Ng, P.-W.: *Aspect-Oriented Software Development with Use Cases*. Addison-Wesley (2005)
11. *jUCMNav website*. University of Ottawa; http://softwareengineering.ca/jucmnav (accessed April 2009)
12. Lencastre, M., Araújo, J., Moreira, A., and Castro, J.: Towards Aspectual Problem Frames: an Example. *Expert Systems Journal*, Vol. 25(1):63-75, Blackwell Publishing (February 2008)
13. Mussbacher, G. and Amyot, D.: Goal and Scenario Modeling, Analysis, and Transformation with jUCMNav. *31st International Conference on Software Engineering (ICSE 2009)*, Companion Volume, Vancouver, Canada (May 16-24, 2009)
14. Mussbacher, G. and Amyot, D.: Heterogeneous Pointcut Expressions. *Early Aspects Workshop at ICSE'09*, Vancouver, Canada (May 18, 2009)
15. Mussbacher, G. and Amyot, D.: On Modeling Interactions of Early Aspects with Goals. *Early Aspects Workshop at ICSE'09*, Vancouver, Canada (May 18, 2009)
16. Mussbacher, G., Amyot, D., and Weiss, M.: Visualizing Early Aspects with Use Case Maps. Rashid, A. and Aksit, M. (Eds.), *Transactions on Aspect-Oriented Software Development III*, Springer, LNCS 4620:105-143 (November 2007)
17. Mussbacher, G., Amyot, D., Araújo, J., Moreira, A., and Weiss, M.: Visualizing Aspect-Oriented Goal Models with AoGRL. *Second International Workshop on Requirements Engineering Visualization (REV'07)*, New Delhi, India (October 15, 2007)

18. Mussbacher, G., Amyot, D., Weigert, T., and Cottenier, T.: Feature Interactions in Aspect-Oriented Scenario Models. *10th International Conference on Feature Interactions (ICFI 2009)*, Lisbon, Portugal (June 11-12, 2009)
19. Mussbacher, G., Amyot, D., Whittle, J., and Weiss, M.: Flexible and Expressive Composition Rules with Aspect-oriented Use Case Maps (AoUCM). *10th International Workshop on Early Aspects (EA 2007)*, Vancouver, Canada (March 13, 2007). Moreira, A. and Grundy, J. (Eds.), *Early Aspects: Current Challenges and Future Directions*, Springer, LNCS 4765:19-38 (December 2007)
20. Mussbacher, G., Whittle, J., and Amyot, D.: Semantic-Based Interaction Detection in Aspect-Oriented Scenarios. *17th International Requirements Engineering Conference (RE 2009)*, Atlanta, USA (August 31 – September 4, 2009)
21. Mussbacher, G.: Aspect-Oriented User Requirements Notation: Aspects in Goal and Scenario Models. Giese, H. (Ed.), *Models in Software Engineering: Workshops and Symposia at MoDELS 2007*, Springer, LNCS 5002:305-316 (August 2008)
22. Mylopoulos, J. and Castro, J.: Tropos: A framework for requirements-driven software development. Brinkkemper, J. and Solvberg, A. (Eds.), *Information Systems Engineering: State of the Art and Research Themes*, Springer, 261-273 (June 2000)
23. Pourshahid, A., Mussbacher, G., Amyot, D., and Weiss, M.: An Aspect-Oriented Framework for Business Process Improvement. *4th International MCeTech Conference on eTechnologies* (MCeTech 2009), Ottawa, Canada (May 4-6, 2009)
24. Rashid, A., Moreira, A., and Araújo, J: Modularisation and Composition of Aspectual Requirements. *2nd International Conference on Aspect Oriented Software Development (AOSD)*, Boston, USA (2003)
25. *URN Virtual Library.* http://www.usecasemaps.org/pub (accessed April 2009)
26. van Lamsweerde, A., Letier, E., and Darimont, R.: Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, Vol. 24(11):908-926, IEEE Press, USA (1998)
27. Whittle, J., Jayaraman, P., Elkhodary, A., Moreira, A., and Araújo, J.: MATA: A Unified Approach for Composing UML Aspect Models based on Graph Transformation. *Transactions on Aspect-Oriented Software Development*, Springer, LNCS, to appear.
28. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *3rd IEEE Int. Symp. on Requirements Engineering*, Washington, USA, IEEE CS, 226-235 (1997)