Heterogeneous Pointcut Expressions

Gunter Mussbacher and Daniel Amyot
SITE, University of Ottawa
800 King Edward
Ottawa, ON, K1N 6N5, Canada
{gunterm | damyot}@site.uottawa.ca

Abstract

Over the last decade, many aspect-oriented (AO) programming and modeling languages have been developed. Pointcut expressions are a key concept of each of these languages as they define the patterns that must be matched for aspects to be applied to the base. To date, most pointcut expressions are constrained to one particular notation – the one for which they were designed – even though a goal of aspect-oriented software development should be to encapsulate a concern through all phases of software development. Motivated by examples of aspects that require characteristics expressed in different notations to be matched by their pointcut expressions, we argue that there should be more focus on heterogeneous pointcut expressions *that can span several notations from potentially different development phases. We demonstrate such pointcuts in an example modeled with the Aspect-oriented User Requirements Notation (AoURN) which combines notations for goal-oriented, scenario-based, and aspect-oriented modeling in one framework for requirements engineering.*

1. Introduction

Many aspect-oriented programming (e.g., [1]) and aspect-oriented modeling techniques (e.g., [2]-[9]) have been introduced over the last decade. Most techniques allow pointcut expressions to be defined for only one modeling notation or programming language. There are many situations, however, which require aspects to be applied over software artifact boundaries as well as across software development phases based on the matching of properties from different models.

For example, requirements may have to be added or removed from requirements documents depending on design decisions encapsulated by aspects. Behavioral and structural views of UML models may have to be synchronized for aspects. The deployment of a system may have an impact on the system's architecture; again this may be encapsulated by an aspect. Last but not least, decision making for feedback and control systems may be modeled with goal models, while the available modes and alternatives may be defined by scenario models – these different model types have to be coordinated and aspects can help with that.

We argue that heterogeneous pointcut expressions (HPEs) are essential and present an approach that allows us to properly manage such pointcut expressions.

In the remainder of this paper, section 2 provides a brief overview of the Aspect-oriented User Requirements Notation (AoURN), which is used to demonstrate HPEs. Section 3 first discusses examples of HPEs drawn from literature and our experience with AoURN. Then, the architecture of the Heterogeneous Pointcut Manager (HPM), enabling HPEs across software development phases and across AO tools, is presented. This is followed by our conclusion and discussion of future work in section 4.

2. Background

The Aspect-oriented User Requirements Notation (AoURN) [10] is an extension of the User Requirements Notation (URN) [11], a standard published recently by the International Telecommunication Union (ITU). 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 (UCM – 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, while UCM models are used for operational requirements, functional requirements, and performance and architectural

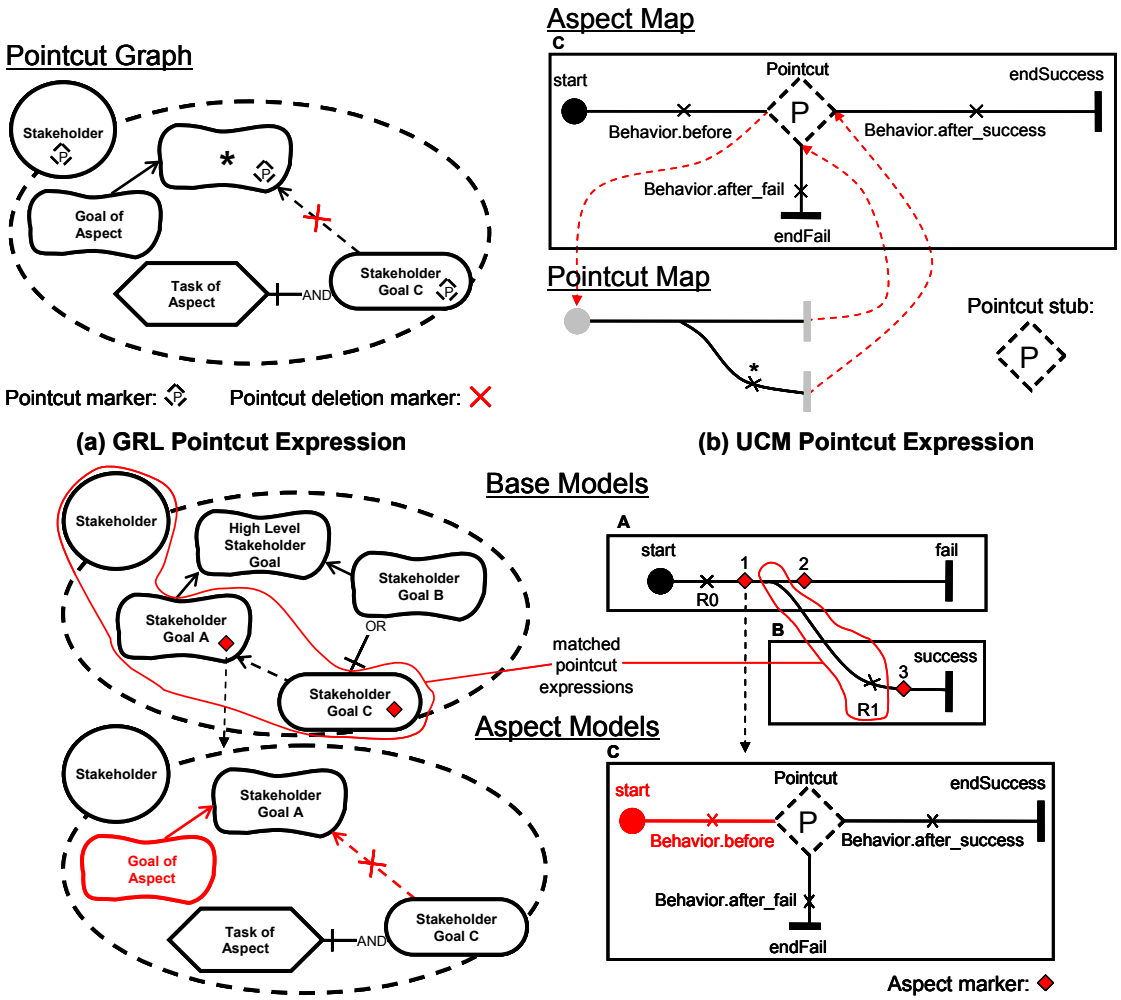

Fig. 1. Aspect-oriented User Requirements Notation

reasoning. In summary, URN has concepts for the specification of stakeholders, goals, non-functional requirements, rationales, behaviour, actors, scenarios, and structuring.

AoURN extends URN with aspect-oriented concepts, allowing modelers to better encapsulate crosscutting concerns which are hard or impossible to encapsulate with URN models alone. AoURN treats concerns as first-class modeling elements, regardless of whether they are crosscutting or not. Typical concerns in the context of URN are a stakeholder's intentions, NFRs, and use cases. AoURN adds aspect concepts to URN's sub-languages, leading to and integrating Aspect-oriented GRL (AoGRL) [12] and Aspect-oriented UCMs (AoUCM) [13][14]. 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 base model or to modify existing elements. AoURN uses standard URN diagrams to describe pointcut expressions (i.e., it is only limited by the expressive power of URN itself as opposed to a particular composition language). Finally, AoURN employs an aspect composition technique that can fully transform URN models.

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.

A UCM model consists of a path that begins at a *start point* (●) and ends with an *end point* (▌). A path may contain *responsibilities* (✕), identifying the steps in a scenario, and notational symbols for alternative (⌯) and concurrent (⊣⊢) branches. Path elements may be assigned to a *component* (□).

AoURN adds the ability to define pointcut expression and then compose aspects with the base model. GRL pointcut expressions are shown on a *pointcut graph* and make use of *pointcut (deletion) markers* (, ✕) to indicate the pattern to be matched (Fig. 1.a). 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. Goals and tasks of an aspect may be described in more detail in separate goal graphs. The top of Fig. 1.c shows the matched pointcut expression in the base model. The locations affected by an aspect are indicated by *aspect markers* (◆). When an aspect marker is selected, the modeler is taken to an aspect view with only those aspectual properties of the pointcut graph highlighted that are relevant to the aspect marker (bottom of Fig. 1.c).

Similarly, UCM pointcut expressions define the pattern to be matched with a *pointcut map* (Fig. 1.b). The aspectual properties, however, are shown on a separate *aspect map*. 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, instead of, in parallel, interleaved, or anything else that can be expressed with the UCM notation). As in Fig. 1.c, the top of Fig. 1.d shows the matched pointcut expression and affected locations with aspect markers (◆), while the bottom shows those aspect properties of the aspect map highlighted that correspond to aspect marker 1.

Note that the matched pointcut expressions are visualized only for illustration purposes but are not part of the concrete syntax of AoURN. Furthermore, the plug-in bindings (the connections between the pointcut stub and the pointcut map) are also visualized only for illustration purposes.

3. Heterogeneous Pointcut Expressions

To the best of our knowledge, there are two examples of pointcut expressions that span more than one modeling notation for a single aspect – we call them *heterogeneous pointcut expressions* (HPEs). In the context of software design with UML, Kienzle, Al Abed, and Klein [15] describe behavioral and structural properties of a UML design in pointcut expressions. Class, sequence, and state diagrams are currently supported and activity diagrams are considered for future work. Pointcut expressions expressed in different notations are synchronized with the help of context bindings and context instantiations.

In the context of requirements engineering with AoURN, Pourshahid, Mussbacher, Amyot, and Weiss [16] describe behavioral and intentional properties of an AoURN model in pointcut expressions. Behavioral properties are expressed in AoUCM (and can therefore easily be extended to structural properties) and intentional properties are expressed in goal models with AoGRL. Pointcut expressions expressed in AoGRL and AoUCM are synchronized with URN links (▶), a URN concept that allows any URN modeling element to be linked with another URN modeling element. While [16] presents a first (and different) example of the general modeling approach discussed here, it does not extrapolate the example to an overall architecture and approach that is applicable to the whole software development process.

A characteristic that applies to both approaches is that the employed notations provide views of the same model and are based on the same metamodel. This simplifies the management and synchronization of aspects with HPEs as well as AO tool support, because an AO tool does not need to cross metamodel boundaries.

Aspects, however, intend to encapsulate concerns through all phases of software development. Inevitably, specialized AO tools are necessary to address the unique needs of various software process activities. Each tool provides composition mechanisms and notations for pointcut expressions that best suit the circumstances. It is unrealistic to assume that all of these tools will be able to directly interoperate with each other, as this, at the minimum, requires transformations or translations between each pair of tools. Furthermore, it is unrealistic to assume that all artifacts created in all phases of software development will share a common metamodel or even a common meta-metamodel. Typically, these artifacts are connected with each other through traceability links. These traceability links, however, are not necessarily part of the software development technique that is used to describe aspects but are rather external to the technique.

Because traceability links and URN links are very similar, we will continue by explaining in more detail how HPEs are supported in AoURN. The AoURN approach is then generalized to concerns that span software development phases and (meta-)metamodels and the architecture of a generic tool in support of HPEs is suggested.

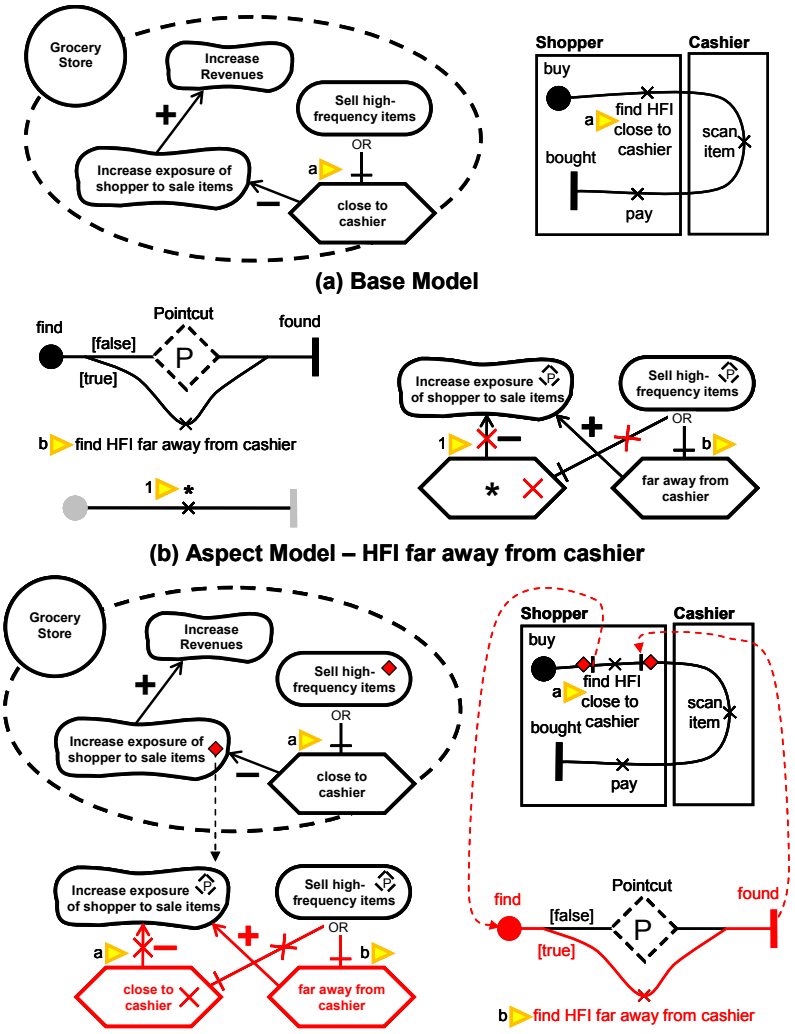

Fig. 2. Heterogeneous Pointcut Expression in AoURN

Fig. 2 depicts a simple example AoURN pointcut expression that spans goal models and scenario models. In the base scenario model (right side of Fig. 2.a), a shopper first finds a high-frequency item (HFI, e.g., milk) in a grocery store close to the cashier, the cashier then scans the item, and the shopper pays the cashier. The base goal model (left side of Fig. 2.a) shows that this is one alternative for the goal of selling HFIs for the grocery store, but that it has a negative impact on the exposure of a shopper to sale items in the store, and therefore, does not increase revenues.

The aspect in Fig. 2.b encapsulates an alternative to this scenario where the shopper finds HFIs far away from the cashier. Note that the [true] and [false] branches are AoUCM's way of modeling replacement – the [false] branch is replaced by the [true] branch. The pointcut expression matches against any responsibility in the base model. However, this responsibility must be linked to a task in the goal model which in turn must be an alternative for selling HFIs and have a negative impact on increasing exposure to sale items in the store. The AoURN aspect therefore encodes the reasons why this aspect should be applied in addition to its behavioral and structural properties.

If the pointcut expression can be matched, the aspect is applied. In the scenario model, the matched responsibility is replaced by the aspect's responsibility. In the goal model, the matched task and its links are removed (because of the pointcut deletion marker) and the aspect's task and its links are added. Fig. 2.c shows the aspect applied to the base model. Note that aspect markers with vertical bars indicate replacement in the base model – one can think of them as entrances (◆|) and exits (|◆) of a tunnel that circumvents the replaced behavior.

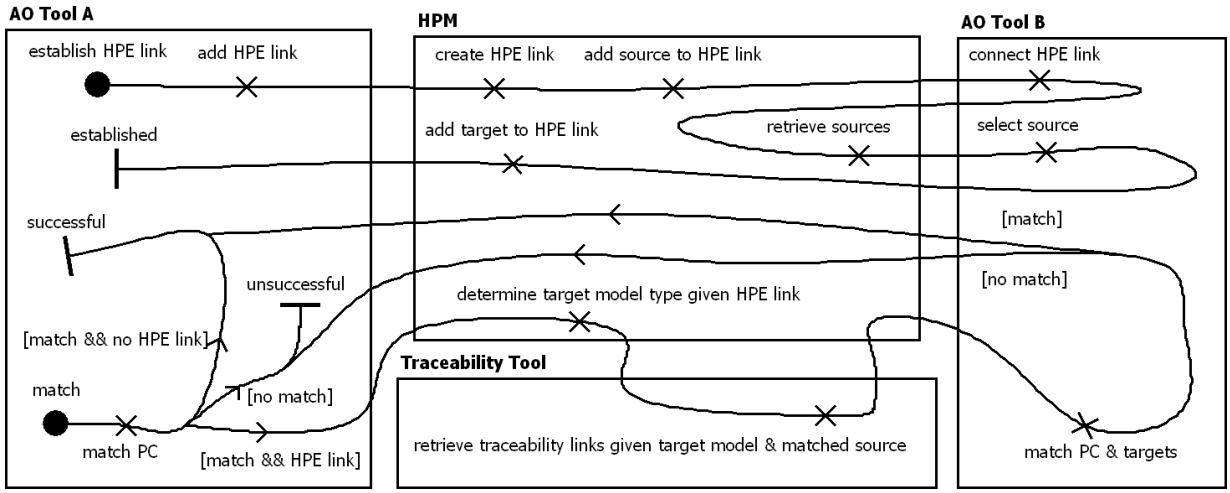

Fig. 3. Architecture of the Heterogeneous Pointcut Manager (HPM)

The URN link that connects the AoUCM and AoGRL pointcut expressions can be thought of as an AND operator. For a successful match of the overall pointcut expression, a match of the AoUCM pointcut expression must have a corresponding match of the AoGRL pointcut expression. For example, the AoUCM pointcut expression in Fig. 2 would match all three responsibilities in the base model, if the URN link is not specified. Therefore, three matches would be reported each containing one of the three responsibilities and the single match of the AoGRL pointcut expression. With the link, however, this is reduced to one responsibility and the AoGRL match.

Generalizing this mechanism leads to an architecture for the *Heterogeneous Pointcut Manager* (HPM), a tool that connects AO tools which define and match pointcut expressions for various modeling notations and traceability tools which manage links between model elements. A prerequisite of this architecture is that each model element in each AO tool can be uniquely identified and thus linked by traceability tools. The UCM in Fig. 3 describes the two main usage scenarios of HPM: establishing links between pointcut expressions and matching pointcut expressions which may contain such links.

If a modeler wants to create an HPE for an aspect in two AO tools, e.g., A and B, an HPE link needs to be established. The modeler therefore adds an HPE link to the pointcut expression in tool A with the help of HPM. HPM creates a new HPE link and adds the element from tool A as the source of the HPE link. Next, the modeler connects the HPE link to the pointcut expression in tool B, by selecting an existing source from the choices given by HPM. HPM in turn adds the element from tool B as the target of the HPE link, hence establishing the HPE link.

During the matching process, the pointcut expression (PC) of tool A is matched first. If there is no match, the process ends unsuccessfully. If there is a match and the pointcut expression does not contain an HPE link, the process ends successfully. If, however, there is an HPE link in the pointcut expression, HPM retrieves all potential traceability links from the traceability tool based on its information about the HPE link. These traceability links must have as the source the matched element from tool A and as a target an element from the same model as the target in the HPE link. HPM then forwards the targets from the retrieved traceability links to tool B. Tool B tries to match the pointcut expression against these targets and the base model in tool B. If no match can be found, the process ends unsuccessfully. If a match is found, the process ends successfully.

Consequently, HPM abstracts from different kinds of traceability tools as it is the only component that needs to interact with the traceability tool. Each AO tool, on the other hand, must only be able to interface with HPM. Hence, an AO tool must be able to define HPE links for pointcut elements and take these links into account during the matching process. Furthermore, an AO tool must provide an API that allows HPM to query the AO tool on whether a match exists given an HPE link and possible targets.

4. Conclusion and Future Work

In this paper, we motivated the need for heterogeneous pointcut expressions (HPEs), presented an example of such pointcut expressions in more detail with the help of the Aspect-oriented User Requirements Notation (AoURN), and introduced the architecture of the Heterogeneous Pointcut Manager (HPM), a tool

that simplifies the management of HPEs. HPEs build on the concept of traceability links, which are typically established between various artifacts of the software development process that may or may not have a common (meta-)metamodel. In the case of AoURN, generic URN links are used to establish HPE links within AoURN models.

While the presented example uses requirements models and a language with built-in support for traceability links, the approach can be generalized to any artifacts that are connected by links, even if the links are external to the models. At this point, HPE links are only interpreted as an AND operator. In future work, we plan to investigate the usage of optional HPE links as well as arbitrarily typed HPE links. We also plan to implement an HPM prototype.

Furthermore, the relationship of heterogeneous pointcut expressions and hybrid pointcuts (as e.g. discussed in [17] in the context of aspect-oriented programming) needs to be further explored.

Acknowledgements. This research was partially supported by NSERC Canada, through its programs of Discovery Grants and Postgraduate Scholarships.

References

[1] *AspectJ web site*. http://www.eclipse.org/aspectj/ (accessed February 2009)

[2] 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.

[3] Chitchyan, R., Rashid, A., Rayson, P., and Waters, R.: "Semantics-Based Composition for Aspect-Oriented Requirements Engineering". *Aspect-Oriented Software Development (AOSD)*, Vancouver, Canada, ACM, 36-48, 2007.

[4] Clarke, S. and Baniassad, E.: *Aspect-Oriented Analysis and Design: The Theme Approach*. Addison Wesley, 2005.

[5] Cottenier, T., van den Berg, A., and Elrad, T.: "Motorola WEAVR: Model Weaving in a Large Industrial Context". *Aspect-Oriented Software Development (AOSD)*, Vancouver, Canada, 2007.

[6] France, R., Ray, I., Georg, G., and Ghosh, S.: "Aspect-oriented approach to early design modeling". *IEE Proceedings - Software,* vol. 151, pp. 173-186, 2004.

[7] Jacobson, I. and Ng, P.-W.: *Aspect-Oriented Software Development with Use Cases*. Addison-Wesley, 2005.

[8] Klein, J., Helouet, L., and Jézéquel, J.-M.: "Semantic-Based Weaving of Scenarios". *Aspect-Oriented Software Development (AOSD)*, Vancouver, Canada, pp. 27-38, 2006.

[9] Whittle, J., Moreira, A., Araújo, J., Jayaraman, P., Elkhodary, A., and Rabbi, R.: "An Expressive Aspect Composition Language for UML State Diagrams". *Model Driven Engineering Languages and Systems, 10th International Conference, MODELS 2007*, Springer, LNCS 4735, pp. 514-528, 2007.

[10] Mussbacher, G.: "Aspect-Oriented User Requirements Notation". Giese, H. (Editor), *Models in Software Engineering: Workshops and Symposia at MODELS 2007*, Springer, LNCS 5002, pp 305-316, August 2008.

[11] ITU-T: *ITU-T Recommendation Z.151 (11/08): User Requirements Notation (URN) - Language definition*. ITU-T, Geneva, Switzerland, November 2008.

[12] Mussbacher, G., Amyot, D., Araújo, J., Moreira, A., and Weiss, M.: "Visualizing Aspect-Oriented Goal Models with AoGRL". *2nd Intl. Workshop on Requirements Engineering Visualization (REV'07)*, New Delhi, India, October 2007.

[13] Mussbacher, G., Amyot, D., and Weiss, M.: "Visualizing Aspect-Oriented Requirements Scenarios with Use Case Maps". *International Workshop on Requirements Engineering Visualization (REV 2006)*, Minneapolis, USA, Sep. 2006.

[14] Mussbacher, G., Amyot, D., and Weiss, M.: "Visualizing Early Aspects with Use Case Maps". Rashid, A. and Aksit, M. (Editors), *Trans. on Aspect-Oriented Software Development III*, Springer, LNCS 4620, pp 105-143, 2007.

[15] Kienzle, J., Al Abed, W., and Klein, J.: "Aspect-Oriented Multi-View Modeling". *8th International Conference on Aspect-Oriented Software Development (AOSD 2009)*, Charlottesville, USA, March 2009.

[16] Pourshahid, A., Mussbacher, G., Amyot, D., and Weiss, M.: "An Aspect-Oriented Framework for Business Process Improvement". *4th International MCETECH Conference on e-Technologies*, Ottawa, Canada, May 2009. LNBIP, Springer (to appear).

[17] D'Hondt, M.: *Hybrid Aspects for Integrating Rule-Based Knowledge and Object-Oriented Functionality*. PhD thesis, Vrije Universiteit Brussel, Belgium, May 2004.