Modeling Aspects in Requirements using SysML Extensions

Kênia Santos de Oliveira and Michel S. Soares
Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil

Keywords: Aspects, Requirements Engineering, SysML.

Abstract: Aspects in software have been proposed and widely studied for the implementation phase of software development to solve modularization issues. Software requirements may also contain scattered and tangled concerns which needs special treatment. The separation of crosscutting concerns at the level of requirements contributes to improve the process of software development, to detect initial conflicts of interest and to improve the modularity of requirements. The purpose of this article is to use SysML to model aspects at the requirements level. This choice was made based on SysML's specific diagram for requirements modeling. Extensions to the metamodel of the SysML Requirements diagram were proposed in order to include aspects during activities of requirements modeling. As a result, for the implementation phase of software development, aspects would have already been identified and modeled during the requirements phase.

1 INTRODUCTION

The activities of identifying and locating interests in well-separated modules is known as separation of concerns (Dijkstra, 1997). In order to obtain modularized software specifications and with the correct partitioning of their concerns, a new paradigm emerged for software development named Aspect Oriented Software Development (AOSD) (Filman et al., 2004), originated from Aspect Oriented Programming (Kiczales et al., 1997). Similar to what was evidenced by the community of Aspect Oriented Programming, requirements may include scattering and entanglement concerns which needs special treatment (Sampaio and Rashid, 2008).

According to (Brito, 2008), with the purpose of conducting an aspect oriented framework in the context of requirements engineering, a number of issues has to be considered. This includes, for instance, support to separation of all interests, including crosscutting concerns, the definition of mechanisms to specify concerns composition and handling conflict situations, and the support to mechanisms to ensure traceability between stages of the development process. An aspect requirement is an interest of the stakeholder that interferes in other requirements or artifacts (Rashid et al., 2006). Provided that aspects at the requirements level have been identified, it is important to represent them, as well as specify their impact and influence on other requirements of the system (Rashid et al., 2006).

In the early stages of software development, it is useful to define some kind of mapping between aspect oriented requirements models and aspect oriented architecture models (Sánchez et al., 2010). However, in many studies this is not considered because aspects are taken into consideration only during implementation. There is still lack of techniques and tools to deal with this issue, especially in establishing a relationship of Requirements Engineering with Software Architecture, which can interfere in the representation of the initial aspects and consequently the identification of aspects in other phases of software development.

Models for representing aspects at early stages of software development have been proposed in the literature (Brito, 2008) (Sánchez et al., 2010) in different domains using UML Use Cases and their scenarios. The main issue with these approaches is that Use Cases are specific to model scenarios of requirements, not individual requirements. The SysML modeling language (OMG, 2012) extends UML with new diagrams, including one specific to model requirements. SysML does not have extensions for modeling aspects, but these can be created as SysML is an extensible modeling language. Thus, the purpose of this study is to use SysML to model aspects at the requirements level. The main reason for choosing SysML is because the language has a specific diagram for modeling individual requirements as well as their relationships.

Santos de Oliveira K. and S. Soares M..
Modeling Aspects in Requirements using SysML Extensions.
DOI: 10.5220/0004419601260133
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 126-133
ISBN: 978-989-8565-60-0

2 REQUIREMENTS MODEL WITH SYSML TO REPRESENT ASPECTS

An interesting characteristic of the SysML Requirements diagram is the possibility of modeling other types of requirements beyond the functional ones (Soares et al., 2011). SysML is an interesting choice as modeling language for requirements as it has a specific diagram to represent requirements. The SysML Requirements diagram can appear in other models in order to represent traces between requirements and the software design. This approach facilitates to represent the aspects relation at the level of requirements with aspects at the level of architecture. This relationship is not well-presented in most studies published in the area of early aspects (Sampaio and Rashid, 2008), (Sánchez et al., 2010).

The original SysML requirements model is extended with new attributes, which considers an extended requirement (represented by the stereotype <<ExtRequirement>>) with six additional attributes, as depicted in Figure 1 (Soares et al., 2011).

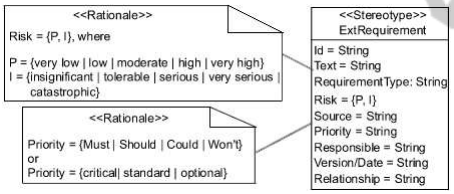

Figure 1: Extended requirement model.

The defined attributes are: **RequirementType**, indicating whether the requirement is functional or non-functional, **Risk**, describing an event or uncertain condition, **Source**, describing where the derived requirement is originated, **Priority**, indicating the order that the requirements should be addressed, **Responsible**, referring to the stakeholder directly responsible for the requirement, **Version/Date**, indicating the requirements version, which is useful to keep track of multiple versions of the requirement, and **Relationship**, which purpose is to improve the activity of tracing requirements to the design models.

The extended model to represent aspects at the level of requirements is presented in Figure 2. The activity of modeling aspects at the level of requirements is based on the created stereotype <<AspectRequirement>> that inherits all the attributes of <<ExtRequirement>>.

Beyond the attributes it is also necessary to define new relationships to represent the composition of aspects. The composition means how an aspect requirement influences or constrains the behavior of a

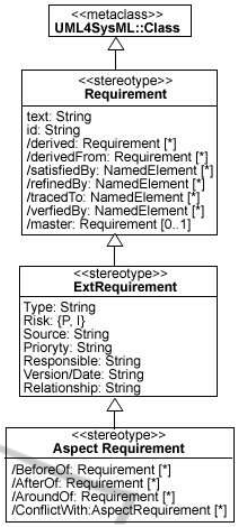

Figure 2: Extended metamodel.

requirement. In order to define these relationships, previously published works (Brito, 2008), (Sánchez et al., 2010) were analyzed. The relationships defined in these studies are equivalent to the relationships *before*, *after* and *around* described in aspect oriented programming. At the architectural level, as well as at the implementation level, the relationships *before*, *after* and *around* are often used to represent the composition of an aspect with other element (Pinto et al., 2011). Because of this characteristic, and to maintain traceability of aspects at the level of requirements to aspects at the architectural level, the relationships <<*before*>>, <<*after*>>, <<*around*>> and <<*conflict*>> are proposed in this article to be applied at the requirements level.

Figure 3 depicts the extended SysML requirement relationship model to represent the aspect requirement relationships. The dashed rectangles are the new relationships.

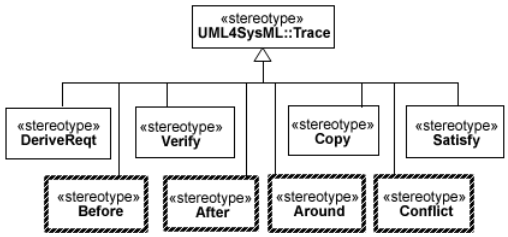

Figure 3: Extended relationships model.

The type *before* means that the aspect requirement is inserted before the requirement. The type *after* means that the aspect requirement is inserted after the requirement. The type *around* can override the behavior of the requirement or part of the aspect requirement can be inserted before and part after the re-

quirement. The relationship <<*conflict*>> indicates whether there is a trade-off between aspects.

Aspects can be represented within a package. Thus, in order to present the relationship between an aspect package and an extended requirement the relationship <<crosscut>> is used. For this, the dependency model was extended as shown in Figure 4. This relationship makes it clear which aspects crosscuts other requirements elements.

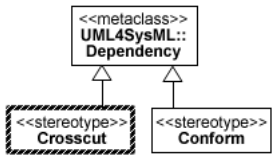

Figure 4: Extended dependency model.

Figure 5 depicts the representation of the relationship <<crosscut>>.

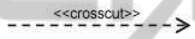

Figure 5: Representation of the crosscut relationship.

3 PROCESS TO REPRESENT ASPECTS AT THE LEVEL OF REQUIREMENTS

The proposed process to represent aspects at the requirements level is described in the Activity diagram depicted in Figure 6. A list of requirements in natural language is the entry artefact in this process.

In stage 1, requirements are separated in accordance with the viewpoints. In SysML, Viewpoints are stereotypes of UML classes. Viewpoints generally represent stakeholders or other systems which have a specific interest in the current system (Sampaio et al., 2007). Initially, only functional requirements are defined. Viewpoints are represented using the viewpoints and view of SysML and also the extended model to requirements (<<ExtRequirement>>).

In stage 2, non-functional requirements are represented separately. The reason is because, according to (Sampaio et al., 2007), usually non-functional requirements are natural candidates to be defined as aspects at the level of requirements engineering, since they are broadly scoped properties which tend to restrict other requirements. The SysML Requirements diagrams with extensions (<<ExtRequirement>>) are used to design these requirements.

In stage 3, non-functional requirements are related with the viewpoints. This is a way to identify which non-functional requirements affect the functional ones, and thus allow to check which are candidates for aspects. This relationship is realized through an extension to the SysML Table. The tabular format facilitates the tracing of requirements during the life cycle. Traceability helps in identifying the source, destination and connections between aspects and requirements. The extended table to relate non-functional requirements with viewpoints is depicted in Table 1, which is composed by the requirement identification (id), the non-functional requirement name, the functional requirement name, and the viewpoint name that the functional requirement pertains. The column relation, which is optional, indicates that there is a relation between the non-functional requirement and the viewpoint.

Table 1: SysML table extended to relate non-functional requirements with viewpoints.

id1	*non-functional requirement*	*relation*	*id2*	*functional requirement*	*viewpoint*

In stage 4, the functional requirements that are repeated in the different viewpoints are identified. These are registered in an extended SysML table as shown in Table 2. This is important to identify the functional requirements which may be candidates for aspects. The column repeat, which is optional, indicates the repetition of functional requirements.

Table 2: SysML table extended to relate functional requirements with functional requirements.

id1	*functional requir.1*	*viewpoint1*	*repeat*	*id2*	*functional requir.2*	*viewpoint2*
			-			

In stage 5, candidate aspects are identified. For non-functional candidates, the identification is done by checking which non-functional requirements are included in more than one viewpoint (the verification is performed by observing table 1). For functional candidates, the identification is realized by checking which functional requirements are repeated in more than one viewpoint (the verification is performed by observing Table 2). Candidate aspects are registered in Table 3. In this table, the relationship column is filled with the types of relationships, which can be *before*, *after* or *around*.

Table 3: SysML table extended to represent the relationship of functional candidates aspects.

id1	*aspect*	*relationship*	*id2*	*functional requirement*	*viewpoint*

In stage 6, the conflicts of interest between aspects are identified. The composition of the interest

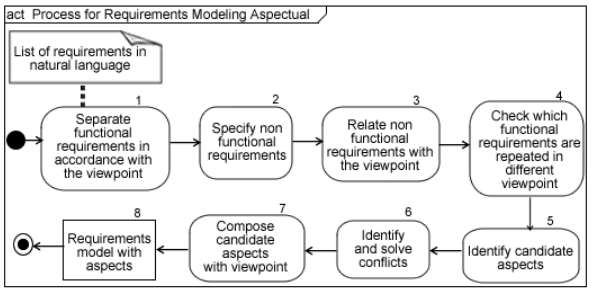

Figure 6: Process for aspect requirements modeling.

may raise conflict situations that need to be identified and resolved. For instance, security and response time may both crosscuts the same requirement of a viewpoint and may contribute negatively to each other. In order to maintain traceability of conflicts, they may also be represented in an extended SysML table as shown in Table 4. The column conflict, which is optional, indicates the conflict between aspects.

Table 4: SysML table extended to represent conflicts between aspect requirements.

id1	*aspect1*	*conflict*	*id2*	*aspect2*	*viewpoint*

In stage 7, candidate aspects are composed with the viewpoints. The tables defined previously already indicate the composition of aspects with requirements. However, at this stage, aspects are composed in the graphical model for representing requirements and for this the extensions for aspects of the SysML Requirements diagram are applied. Finally, in stage 8 the requirements model with aspects is obtained.

4 TOOL SUPPORT

The ArgoUML (0.35.1) (Ramirez et al., 2011) software tool was extended with the metamodel proposed in this article in order to support the modeling of requirements and aspect requirements. In order to perform the modifications in the source code of ArgoUML, the technique of Code Clone (Baker, 1995) was applied. The comprehension of the source code started with the debug process in the software development tool Eclipse (version Juno). From the debug process, it was possible to locate some classes that implement the class element of ArgoUML. Thus, all the code that builds the class element was cloned and modified for the construction of the new elements.

A visualization of the ArgoUML tool with the proposed extensions is depicted in Figure 7. The arrows drawn in the figure indicate the buttons of new elements inserted in the tool. In the modeling environment an example of each element is represented. The illustrated elements are *<<Block>>*, *<<Aspect Block>>*, *<<Requirement>>* and *<<Aspect Requirement>>*. The element *<<Block>>* has compartments defined by SysML (*constraints*, *parts*, *references*, *values*, *properties* and *operations*). The element *<<Aspect Block>>* has, beyond the compartments defined by SysML, also compartments defined in this work (*point* and *advice*).

5 CASE STUDY

The Health Watcher (HW) system was used as a case study. HW has been used as a reference for the development of aspect oriented software because of the heterogeneity of crosscutting concerns encountered in its implementation (Chavez et al., 2009).

HW is an information system based on Web platform developed to improve the quality of services offered by the Department of Health of a city hall (Massoni et al., 2006). The full document of the original description of requirements can be obtained in (Massoni et al., 2006). Figure 8 depicts the viewpoint Employee. This viewpoint is one among all viewpoints obtained in the case study. In the second step, the non-functional requirements are modeled. As an example, Figure 9 shows the non-functional requirement *Usability*.

In the third step, the non-functional requirements are related to each viewpoint using the SysML extended table. For instance, in Table 5, the relationship of the non-functional requirement *Usability* with *viewpoints* is described.

In the fourth step, the functional requirements that appears in multiple viewpoints, i.e., are repeated, are identified. In this case study, only the login requirement is repeated in different viewpoints, as shown in Table 6.

In the fifth step, candidate aspects are identified,

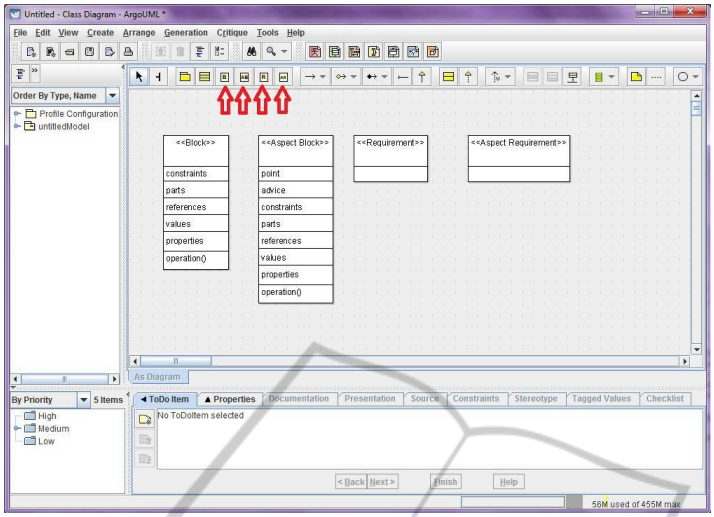

Figure 7: Visualization of ArgoUML tool with extensions.

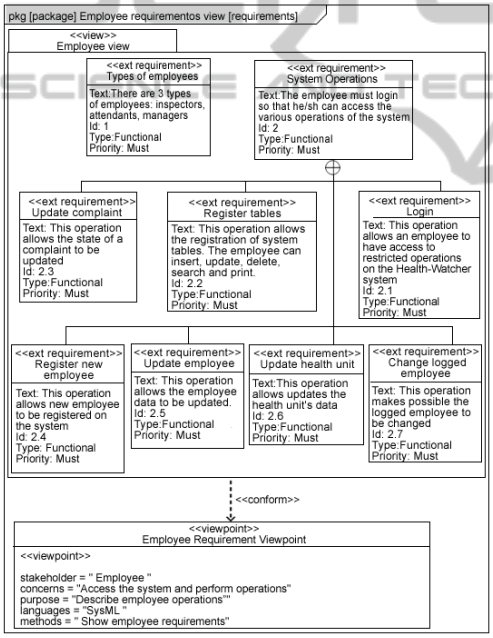

Figure 8: Viewpoint employee.

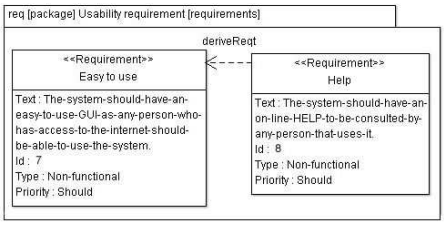

Figure 9: Non-functional requirement usability.

as shown in Table 7. In case the requirement participates in more than one viewpoint, then one of the identifiers (id) is chosen.

Table 5: Relationship table of the non-functional requirements with viewpoints.

id1	*non-functional requirement*	*relation*	*id2*	*functional requirement*	*viewpoint*
7	Easy to use	-	2	System Operation	Employee
7	Easy to use	-	4	Interact with the system	Citizen
7	Easy to use	-	5	Search	Citizen
7	Easy to use	-	6	Register a complaint	Citizen
8	Help	-	2	System Operation	Employee
8	Help	-	4	Interact with the system	Citizen
8	Help	-	5	Search	Citizen
8	Help	-	6	Register a complaint	Citizen

Table 6: Table to relate functional requirements with functional requirements.

id1	*functional requir.1*	*viewpointt1*	*repeat*	*id2*	*functional requir.2*	*viewpoint2*
2.1	Login	Employee	-	6.1	Login	Citizen

In the sixth step, conflicts of interest between aspects are identified. In this case study, there is a possibility of conflict between “Security Protocol” and “Response Time” because the security implementation may compromise the response time. The solution to identified conflicts such as this one can be further negotiation with the stakeholders. In the seventh step, candidate aspects are composed with viewpoints in the graphical model as shown in Figure 10.

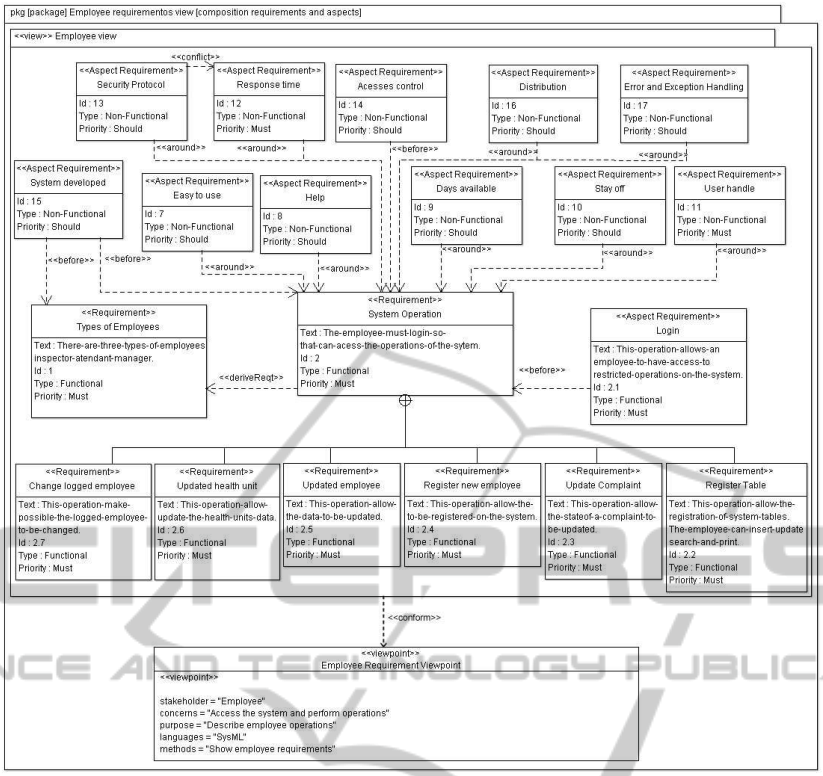

Figure 10: Composition of aspects and requirement in the viewpoint employee.

Table 7: Table to represent the relationship of candidate aspects with viewpoints.

id1	*aspect*	*relationship*	*id2*	*functional requirement*	*viewpoint*
7	Easy to use	before	2	System Operation	Employee
7	Easy to use	before	4	Interact with the system	Citizen
7	Easy to use	before	5	Search	Citizen
7	Easy to use	before	6	Register a complaint	Citizen
8	Help	around	2	System Operation	Employee
8	Help	around	4	Interact with the system	Citizen
8	Help	around	5	Search	Citizen
8	Help	around	6	Register a complaint	Citizen
2.1	Login	before	2	System Operations	Employee
2.1	Login	before	6	Register a Complaint	Citizen

Table 8: Conflicts between aspect requirements.

id1	*aspect1*	*conflict*	*id2*	*aspect2*	*viewpoint*
16	Security Protocol	-	15	Response Time	Employee

6 COMPARISON WITH OTHER MODELS

In order to evaluate the proposed model in this work, a comparison with other five studies is proposed in this section. These studies were chosen because they are among the most cited works in this field. The chosen works are:

A Modularization and Composition of Aspectual Requirements (Rashid et al., 2003).

B Theme: An Approach for Aspect-Oriented Analysis and Design (Baniassad and Clarke, 2004).

C Scenario Modelling with Aspects (Whittle and Araujo, 2004).

D Semantics-based Composition for Aspect-Oriented Requirements Engineering (Chitchyan et al., 2007).

E Mining Aspects in Requirements (Sampaio et al., 2005).

F Modeling Aspects at the Requirements Level with SysML (this article)

Table 9 summarizes the comparison. The symbols (and their semantics) used for evaluation are as follows:

■ Evaluated criteria is fully satisfied.

Table 9: Comparison between models.

Criteria	A	B	C	D	E	F
Process to identify crosscutting requirements	■	■	■	□	■	■
Identification of functional and non-functional crosscutting requirements	⊡	■	⊡	□	□	■
Composition of aspects and requirements	■	□	■	■	□	■
Identification of Conflicts	■	□	■	■	□	⊡
Resolution of Conflicts	■	□	□	□	□	□
Graphical modeling	□	■	■	□	□	■
Specific diagram for requirements	□	□	□	□	□	■
Relationship between requirements	□	□	□	□	□	■
Relationship with UML	□	□	■	□	□	■
Extensibility of the Model	■	□	■	■	□	■
Relationship with Architecture	■	□	□	□	□	■
Traceability with design	■	■	□	□	□	□
Tools support	□	⊡	⊡	⊡	⊡	⊡

⊡ Evaluated criteria is partially satisfied.

□ Evaluated criteria is not satisfied.

The model proposed in (Rashid et al., 2003) presents a set of composition rules that defines the relationship between requirements and aspectual requirements. However, the model does not propose to represent the relationship between functional requirements. A process to identify and solve conflicts is proposed by using a table which describes how a requirement affects another one. In order to solve conflicts, weights from 0 (less important) to 10 (most important) to conflicting aspect requirements are proposed. The model is based on XML and is purely textual.

The model presented in (Baniassad and Clarke, 2004) is based on natural language. A semi-automatic computer tool is used to identify crosscutting requirements in specifications. The identification process is based on a lexical analysis in requirements documents, using specific keywords provided by stakeholders. One result provided by the tool is a graphic view of the aspects. The proposed model does not provide means to solve conflicts.

The main idea behind the model proposed by (Whittle and Araujo, 2004) is to compose requirements and scenarios. Aspects are modeled using a language based on UML, and scenarios are modeled using UML Sequence diagrams and Use Cases. The identification of aspects is based on verifying which Use Cases are influenced by non-functional requirements. The identification of conflicts is performed, but it is not described in the paper. The final model does not relates to software design or architecture.

A requirements description language is described in (Chitchyan et al., 2007) with the purpose of improving the requirements specification written in natural language with additional semantic details. A process to identify aspects is not described.

The main purpose of the approach presented in (Sampaio et al., 2005) is to identify candidate aspects in unstructured requirements documents. An automatic tool helps the developer to mine and to model crosscutting concerns. The tool is customized, with the possibility of identifying crosscutting concerns in different ways. The tool makes searches in the specification and produces a model that represents the relationship between requirements. However, the work does not propose a model to compose or identify conflicts.

The model proposed in this work makes use of a defined process to identify functional and non-functional aspects. The new proposed relationships allows the composition of requirements with aspectual requirements. An approach to solve conflicts is also proposed with priority levels. The graphical modeling is described using an extension of the SysML Requirements diagram.

7 CONCLUSIONS

This article proposes an approach to identify and to handle crosscutting concerns at the requirements level using the SysML modeling language. The process consists of seven steps using the resources of the SysML modeling language with proposed extensions to its metamodel. The SysML viewpoints are used in stage 1 and the requirements are modeled using the SysML Requirements diagram in stage 2. By the end of stages 3, 4, 5 and 6, information of requirements and aspects modeled are stored into an extended SysML Table. Finally, in stage 7, conflicts such as relationships between requirements are solved. The proposed model supports the separation of all crosscutting concerns and mechanisms to spec-

ify the composition of concerns and handling conflict situations. With the proposed model, for the implementation phase of software development, aspects have already been identified and modeled during the requirements phase. The proposed model supports traceability of aspects at the architectural level with the SysML proposed relationships. The tables used in the process to represent aspects requirements facilitates the tracing of requirements during the life cycle of the system. A tool support is provided with the proposed model implemented in an extension of the ArgoUML tool.

ACKNOWLEDGEMENTS

The authors would like to thank CNPq (www.cnpq.br) for the financial support.

REFERENCES

Baker, B. S. (1995). On Finding Duplication and Near-Duplication in Large Software Systems. In *Proceedings of the Second Working Conference on Reverse Engineering*, WCRE '95, pages 86–95.

Baniassad, E. and Clarke, S. (2004). Theme: An Approach for Aspect-Oriented Analysis and Design. In *Proceedings on the 26th International Conference on Software Engineering*, pages 158–167.

Brito, I. S. S. (2008). *Aspect-Oriented Requirements Analysis*. PhD thesis, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa.

Chavez, C., Garcia, A., Batista, T., Oliveira, M., Sant'Anna, C., and Rashid, A. (2009). Composing Architectural Aspects Based on Style Semantics. In *Proc. of the 8th ACM International Conference on Aspect-Oriented Software Development*, pages 111–122.

Chitchyan, R., Rashid, A., Rayson, P., and Waters, R. (2007). Semantics-Based Composition for Aspect-Oriented Requirements Rngineering. In *Proceedings of the 6th International Conference on Aspect-Oriented Software Development*, pages 36–48.

Dijkstra, E. W. (1997). *A Discipline of Programming*. Prentice Hall, Upper Saddle River, NJ, USA, 1st edition.

Filman, R. E., Elrad, T., Clarke, S., and Aksit, M. (2004). *Aspect-Oriented Software Development*. Addison Wesley Professional.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J., and Irwin, J. (1997). Aspect-Oriented Programming. In *Proc. of the 11th European Conference on Object-Oriented Programming*, pages 220–242.

Massoni, T., Soares, S., and Borba, P. (2006). Requirements Document Health-Watcher. Technical report. version 2.0.

OMG (2012). *Systems Modeling Language (SysML) Specification - version 1.3*.

Pinto, M., Fuentes, L., and Troya, J. M. (2011). Specifying Aspect-Oriented Architectures in AO-ADL. *Information and Software Technology*, 53:1165–1182.

Ramirez, A., Vanpeperstraete, P., Rueckert, A., Odutola, K., Bennett, J., Tolke, L., and van der Wulp, M. (2011). *ArgoUML User Manual - A tutorial and reference description*.

Rashid, A., Garcia, A., and Moreira, A. (2006). Aspect-oriented Software Development Beyond Programming. In *Proc. of the 28th International Conference on Software Engineering*, pages 1061–1062.

Rashid, A., Moreira, A., and Araújo, J. (2003). Modularisation and Composition of Aspectual Requirements. In *Proceedings of the 2nd international conference on Aspect-oriented software development*, AOSD '03, pages 11–20.

Sampaio, A., Loughran, N., Rashid, A., and Rayson, P. (2005). Mining Aspects in Requirements. In *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop*.

Sampaio, A. and Rashid, A. (2008). Mining Early Aspects from Requirements with Ea-Miner. In *Companion of the 30th International Conference on Software engineering*, pages 911–912.

Sampaio, A., Rashid, A., Chitchyan, R., and Rayson, P. (2007). EA-Miner: Towards Automation in Aspect Oriented Requirements Engineering. In *Transactions on aspect-oriented software development III*, pages 4–39. Springer-Verlag, Berlin, Heidelberg.

Sánchez, P., Moreira, A., Fuentes, L., Araújo, J. a., and Magno, J. (2010). Model-driven Development for Early Aspects. *Information and Software Technology*, 52(3):249–273.

Soares, M. S., Vrancken, J., and Verbraeck, A. (2011). User Requirements Modeling and Analysis of Software-Intensive Systems. *Journal of Systems and Software*, 84(2):328–339.

Whittle, J. and Araujo, J. (2004). Scenario Modelling with Aspects. *IEE Proceedings Software*, 151(4):157–171.