Towards an Integrated AORE Process Model for Handling Crosscutting Concerns

Narender Singh
Department of Computer Science & Applications,
Maharshi Dayanand University,
Rohtak, India.

Nasib Singh Gill
Department of Computer Science & Applications,
Maharshi Dayanand University,
Rohtak, India.

ABSTRACT
The two fundamental principles in software engineering to deal software complexity are *separation of concerns* and *modularity*. A lot of techniques exist in literature adopting these fundamental principles and some success in this direction has been achieved. Despite this improvement, still complete separation of concerns is not achieved and is far from adequate.

Aspect-Oriented Software Development is another step towards achieving improved modularity and aims to advanced separation of concerns. It handles crosscutting concerns in an efficient manner by encapsulation them in separate modules called aspects and further uses composition mechanism to integrate them with core concerns. Handling crosscutting concerns in the early stages of software development is beneficial rather than handling them in later stages of software development because it not only makes the design simpler, but also helps to reduce the cost and defects that occur in the later stages of development.

Aspect-Oriented Requirements Engineering (AORE) focuses on identifying, analyzing, specifying, verifying, and managing the crosscutting concerns at the early stages of software development. In last few years, many researchers contributed their significant efforts in this area but, still it is not sufficient.

In this paper, we have proposed such an approach that incorporates aspect-oriented concepts and which includes concern management as a key separate activity that is not clearly mentioned earlier in literature. Also, traceability is an essential activity to accommodate changes in requirements but it is very difficult to implement. Organizing large numbers of requirements into meaningful and more manageable groups and negotiating specification with clients can make traceability easier to implement and maintain. The proposed approach supports identification, management, specification, and composition of all concerns.

Keywords
Separation of concerns, crosscutting concerns, aspect-oriented software development, aspect-oriented requirements engineering.

1. INTRODUCTION
Over the last decade, software is evolving at a rapid pace along with more advanced technologies. Developing complex software systems is not a straight forward process but it includes implementation of high level concepts and techniques. A large number of projects fail and result into *software crisis* [1] because they violate some constraints. The s*oftware engineering* [2] term was introduced in the NATO Working conference on software engineering in 1968 to cope with *software crisis*. Although hundreds of authors have developed personal definitions of software engineering, a definition proposed by *Fritz Bauer* at the seminal conference on the subject still serves as a basis for discussion: "*Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines*".

The two fundamental principles in software engineering to deal software complexity are *separation of concerns* and *modularity* [3] [4] [5]. A *concern* [6] is any matter of interest in a software system. S*eparation of concerns* [SOC] means that handle one property of a system at a time. In others words, it is quite easy and simple to divide the complex task into smaller subtasks and then handle one by one rather than handling the whole complex task. The smaller subtasks are handled separately and finally integrated for the complete solution. *Modularity* [3] [7] is the principle to structure software into modules where modules are self-contained, cohesive building blocks of software. Each module is responsible for handling certain concerns of the software system and all modules of a software system grasp all the concerns of the system.

A lot of techniques exist in literature which follows these fundamental principles of software engineering. Some success in this direction has been achieved. But, still complete separation of concerns is not achieved even in today's most popular programming paradigm like object-oriented programming. In these traditional techniques, some concerns may be easily encapsulated with their building blocks such as classes, modules, procedures etc. But, same is not possible for another. They are non-modular and spanning over multiple classes, modules, or procedures in a software system and are therefore called *crosscutting concerns*. Typical examples are persistence, logging, exception handling, synchronization, auditing, security etc. Due to their vary nature, crosscutting concerns are responsible for *scattering*, which means the implementation of a concern is spread over several program modules; and *tangling* which means a program module implements multiple concerns.

Aspect-Oriented Software Development (AOSD) [8] is another step towards achieving improved modularity and aims at alleviating the problems of scattering and tangling. It aims at addressing crosscutting concerns by providing means for their systematic identification, separation, representation and composition [9]. Crosscutting concerns are encapsulated in separate modules, known as *aspects*, so that localization can be promoted. It later uses composition mechanism to weave them with other core modules at loading time, compilation time, or run-time [10]. This results in better support for modularization hence reducing development, maintenance and evolution costs.

AOSD was first introduced at programming level, where aspects are handled in code. Many aspect-oriented programming approaches have been proposed such as AspectJ, AspectC, AspectC++, JBoss AOP, JAsCo, HyperJ, adaptive programming, and composition filters. A lot of work also has been carried out at the design level mainly through extensions to the UML meta-model e.g. [11] [12]. Research on the use of aspects at the requirements engineering stage is still immature. Handling crosscutting concerns in the early stages of software development is beneficial rather than handling them in later stages of software development because it not only makes the design simpler, but also helps to reduce the cost and defects that occur in the later stages of development. Aspect-Oriented Requirements Engineering (AORE) focuses on identifying, analyzing, specifying, verifying, and managing the crosscutting concerns at the early stages of software development. It does not replace but rather complements any of the existing requirements methodologies.

In this paper, we attempt to briefly discuss the strengths and shortcomings identified from the literature of traditional requirements engineering approaches and research conducted during last few years in the area of AORE. The main objective of this paper is to address these shortcomings and outline an integrated AORE process model for handling crosscutting concerns.

2. LITERATURE REVIEW

A lot of approaches exist in the literature for separation of concerns at early stages of software development. We generally categorize these approaches as non aspect-oriented and aspect-oriented approaches. Non aspect-oriented approaches include Viewpoint-oriented approaches (PREview and VIM), Goal-oriented approaches (NFRF, I*, and KAOS), Use Cases and Scenario Based Approaches (Use cases, and negative scenarios), and Problem Frames. Aspect-oriented approaches include AORE with arcade, Aspects in Requirements Goal Models, Aspect-Oriented Software Development with Use Cases, Scenario Modeling with Aspects, Aspectual Use Case Driven Approach, Concern Modeling with Cosmos, Concern-Oriented Requirements Engineering, Aspect-Oriented Requirements Engineering for Component-Based Software Systems, and Theme approach.

In last few years, many researchers contributed their significant work in this area for the identification, specification and composition of aspects at requirements engineering phase. In this section, we briefly discuss these efforts. In paper [13], an approach called Aspect-Oriented Component Requirements Engineering (AOCRE) is proposed that focuses on identifying and specifying the functional and non-functional requirements relating to key aspects of a system each component provides or requires. It helps in improving reusability, extensibility, and allocating the responsibility among reused and application-specific components. But, this approach is too specific for component development and there is no evidence of its use in software development in general. Also, the identification of aspects is not clearly defined and lacks tool support. In paper [14], a general model for aspect-oriented requirements engineering is presented that supports identification and separation of concerns at early stages and their mapping and influence on later stages of development. But, it needs more efforts on aspect validation, aspect composition, and conflicts resolution. In paper [15], theme approach is discussed that is divided into two parts: Theme/Doc and Theme/UML. Theme/Doc provides views and functional support for identification at requirements phase, whereas Theme/UML provides standard UML to model each theme and their integration at the design phase. This approach supports identification and analysis of aspects at requirements phase, but does not provide any mechanism for specifying and compositing aspects in a systematic way. Also, this approach is only applicable for structured requirements document. Hence, results in costly and time-consuming for handling large and unstructured requirements documents. In paper [16], a process model for modelling and composition of aspectual scenarios is presented. The approach provides ability to independently specify both aspectual and non-aspectual scenarios and further their composition along with better modularization and traceability. But, the approach lacks on providing any systematic technique for identifying aspectual scenarios and handling conflicts during composition. In paper [17], an approach called aspect-oriented software development with use cases is presented; which is an extension to the traditional Use Case approach proposed by the same author. The approach supports the identification, specification, analysis, design, and implementation of use cases; but does not support conflicts resolution. In paper [18], an approach called concern-oriented requirements engineering that handles both functional and non-functional requirements in a uniform fashion and concerns are not classified as viewpoints, use cases or aspects. This approach supports multi-dimensional separation of concerns too. Hence, avoids the tyranny of dominant decomposition. It also supports early trade-offs and conflicts resolution. But, it lacks on the relationships between two concerns, implementation of proposed model with more case studies for validation. In paper [19], an integrated approach to support Aspectual Requirements is presented which include a process model with a tool. This approach facilitates better understanding of concepts and their analysis along with tool support of identification, specification and composition of concerns. But, the approach lacks in providing any technique to support traceability as well as trade-offs. In paper [20], AOP is applied at early phase of software development i.e. at requirements phase and an approach called aspect-oriented requirements modelling is proposed; which aims to apply AOP at early stages of software development. The approach provides mechanism to cater both functional and non-functional concerns and their representation in class diagrams and state-chart diagrams respectively. Here, core concerns are considered as functional and crosscutting concerns are treated as non-functional. But, a crosscutting concern may also be functional [21]. Hence, this approach does not clearly identify and specify crosscutting concerns which are functional. But it does not mention any method for identifying aspects along with their validity. It also lacks on mapping requirements and traceability. In paper [22], an aspect-oriented approach is presented, which supports separation of crosscutting concerns in activity modelling. This approach is inspired from the Theme/Doc Approach and EA-Miner. In paper [23], a model for identifying crosscutting concerns and supporting it an automated tool is presented. The model is based on Theme/Doc and Early Aspects Identification approaches. This model provides full automatic facility for identifying crosscutting concerns at early stages of software development. But, the model does not provide any mechanism for resolving conflicts and also not provide evidences of its validation using case studies. In paper [24], an approach called Aspect-oriented User Requirements Notation (AoURN) is proposed which integrates goal-based, scenario-based, and aspect-based concepts in a single framework. It clearly represents relationship among goal and scenario models and also provides an ability to manage crosscutting concerns among them. It also supports traceability with the help of pointcut and aspects. It also supports composition mechanism, but only limited to User Requirements Notation (URN). Further, its qualitative as well as quantitative assessment with respect to

AORE models properties and metrics respectively still needed and advanced research should be investigated for its applicability. In paper [25], a model called aspect-oriented software development is proposed, which represents aspects as Use Case Model and Sequence Diagrams from the early stages of software development. The model supports the aspect representation throughout the software development lifecycle. It also supports aspects traceability. The first limitation of this model is that it only works with structured documents i.e. software requirements specifications (SRS). It does not provide any systematic method to handle conflicts. It also lacks on implementing the model and validation it with some case studies. In paper [26], an approach is proposed for identifying aspects from functional and non-functional requirements using system structure. This approach is good in aspect identification but lacks on tools to fully automate the process and also lacks on implementing the model and validation it with some case studies. In paper [27], an approach called use case and non-functional scenario template-based approach is proposed, which is based on use cases. It supports aspects handling along with improved traceability. But, it does not support any formal method and its application in real systems.

3. FLAWS ENCOUNTERED IN TRADITIONAL RE APPROACHES

A lot of traditional (non-AO) requirements engineering approaches exist in literature like Use Cases [28], Goals [29], and Viewpoints [30]. All these approaches identify and treat the requirements in good manners. But, they all fail in handling the crosscutting concerns clearly. The major motivation factor in the invention of AORE approaches is to remove some of the flaws encountered in traditional requirements engineering approaches which are discussed here as:

These approaches suffer from *tyranny of dominant decomposition* symptom i.e. they are modularized in only one way by considering only one type of concerns such as use cases, viewpoints and goals at a time. And the other kinds of concerns that do not align with that modularization result as scattered and tangled modules. For example, PREview has focused on non-functional concerns. On the other hand, Use Cases have focused on functional requirements. In contract, Aspect-Oriented approaches, such as CORE treat all types of concerns equally and consistently. Thus, the first flaw encountered that AORE addressed is the equal treatment of all types of concerns of importance simultaneously.

Some traditional approaches have identified non-functional requirements as crosscutting requirements. But, they do not consider functional requirements as so. Also, crosscutting requirements are not modularized separately. In contract, Aspect-Oriented approaches, such as CORE have considered this issue. Thus, the second flaw encountered that AORE addressed is to identify and characterize the crosscutting influence for both functional and non-functional requirements and modularize them separately as aspects.

Mostly, all the traditional approaches lacks on composition mechanism. AORE provides composition as primary issue and handles it using joinpoint model and composition semantics. The joinpoint model exposes structured points through which requirements can be composed. The composition semantics provide systematic meaning to the composition. Thus, the third flaw encountered in traditional requirements engineering approaches that AORE addressed is lacking the mechanism for requirement composability.

4. A CRITICAL ANALYSIS OF AORE APPROACHES IN LITERATURE

A critical analysis of existing AORE approaches in literature is given in Table 1. Here, an attempt is made to find out strength as well as shortcomings of each approach explained in literature review section of this paper.

Table 1. A critical analysis of AORE approaches

Techniques	Strengths	Shortcomings
Aspect-Oriented Component Requirements Engineering (AOCRE)	-Supports separation of crosscutting functional and non-functional properties. -Improves reusability, extensibility, and allocates the responsibility among reused and application-specific components. - Basic tool support for aspect-oriented requirements engineering.	-Too specific for component development and there is no evidence of its use in software development in general. -Lacks on clearly defining the identification of crosscutting concerns. -Lacks on tool support.
Early-aspects	-Supports separation of crosscutting functional and non-functional properties. -Identify conflicts and establish possible tradeoffs early on in the development cycle. -Improves modularisation and traceability. -Applicable for structured and unstructured requirements document.	-Lacks on validation of aspects, their composition with other requirements and resolution of possible conflicts resulting from the composition process. -Lacks on a notation to describe aspects, their interactions and composition relationships at the requirements level.
Theme	-All the relationship between the requirements are clearly identified and mapped. -Ambiguity in the requirements can be identified. -Supports traceability from requirements to design. -Provides a tool support to create relationships between concerns and the	-This approach is only applicable for structured requirements document. -The developer must posses the domain knowledge. -The developer has to manually map the relationship between the themes and requirements. -It is costly and time consuming to handle large amount of requirement

	requirements.	sources.
Aspect-Oriented Scenario Modeling	-Provides ability to independently specify both aspectual and non-aspectual scenarios. -Also provides composition along with better modularization and traceability.	-Does not provide any systematic technique for identifying aspectual scenarios and handling conflicts during composition. -Lacks on tool support.
Aspect-Oriented Software Development with Use Cases	-Handles both functional and non-functional requirements in a uniform fashion. -Supports the identification, specification, analysis, design, and implementation of use cases.	-Does not support conflicts resolution.
Concern-Oriented Requirements Engineering	-Handles both functional and non-functional requirements in a uniform fashion. -Supports multi-dimensional separation of concerns, early trade-offs and conflicts resolution.	-Lacks on the relationships between two concerns. -Also lacks on implementation of proposed model with more case studies for validation.
Integrated Approach for Aspectual Requirements	-Support better handling of separation, modularization, representation and composition of concerns. -Facilitates better understanding of concepts and their analysis along with tool support.	-Lacks on providing any technique to support traceability as well as trade-offs.
Aspect-Oriented Requirements Modelling	-Provides mechanism to cater both functional and non-functional concerns. -Here, core concerns are considered as functional and crosscutting concerns are treated as non-functional.	-Does not clearly identify and specify crosscutting concerns which are functional. -Also, lacks on mapping requirements and traceability.
Crosscutting Concern Identification using NLP	Provides full automatic facility for identifying crosscutting concerns at early stages of software development.	Does not provide any mechanism for resolving conflicts and also not provide evidences of its validation using case studies.
Aspect-oriented User Requirements Notation (AoURN)	-Integrates goal-based, scenario-based, and aspect-based concepts in a single framework. -Clearly, represents relationship among goal and scenario models and also provides an ability to manage crosscutting concerns among them. -Supports traceability with the help of pointcut and aspects.	-It also supports composition mechanism, but only limited to User Requirements Notation (URN). -Further, its qualitative as well as quantitative assessment with respect to AORE models properties and metrics respectively still needed and advanced research should be investigated for its applicability.
Use case And Non-functional Scenario Template-Based Approach to Identify Aspects	-Supports aspects handling at requirements level. -Improves modularity in the requirements. -Also improves traceability from requirement analysis level to implement level.	-Does not support any formal method and its application in real systems. -Lacks on validation of aspects, their composition with other requirements and resolution of possible conflicts resulting from the composition process. -Lacks on tool support.
Identification of Crosscutting Concerns with UML	It is a simple method and applicable to small scale requirements	It can only identify non-functional crosscutting concerns.
Aspect-Oriented Use Case Modelling	-Handles both functional and non-functional requirements. -Treats business rules as an important source of aspects when it cross-cuts more than one use cases. -Further, improves the reusability of both the base model and the representation of business rules. -Successfully, realizes the composition of aspects to the core functionalities at the requirement level.	-Does not support conflicts resolution. -Lacks on providing any technique to support traceability as well as trade-offs. -Lacks on tool support.

5. MOTIVATION FOR AN INTEGRATED AORE PROCESS MODEL

As mentioned above, the flaws encountered in traditional requirements engineering approaches should be avoided by AORE process models. A lot of work in the area of AORE has been done with some success, but this work is not sufficient to handle all the flaws encountered. Some AORE process models still need to avoid the tyranny of dominant decomposition because they are extensions of existing requirements engineering approaches and result in scattering and tangling; to improve the ability to identify and specify both functional and non-functional crosscutting concerns because most of the approaches consider only non-functional concerns as crosscutting; to manage concerns and to offer a systematic method to handle conflicts. Thus, the main motivation factor behind this work is the analysis of existing AORE approaches and incorporating their strengths into a single integrated approach along with including activity like concern management. Now the question is *why we need concerns management*? If we look at some reports like Standish Group in 1995 [31] and CHAOS Summary 2009 [32]; evidence suggests that most of the projects failures are due to requirements. Hence, for avoiding more projects failures, we need management of requirements and concerns.

6. A PROPOSAL FOR AN INTEGRATED AORE PROCESS MODEL

Our proposal to Integrated AORE process model consists of the following activities as identify concerns, manage concerns, specify concerns, identify crosscutting concerns, and compose concerns.

6.1 Identify concerns

The first phase in our proposed process model is to identify all concerns of the system. A *concern* is any matter of interest in a software system and may represent any feature that the system must have or a constraint that it must satisfy to be accepted by the stakeholders. A requirement in requirements specification document may represent a concern [33]. Generally, we classify requirements into two categories, *functional requirements* and *non-functional requirements.* Functional requirements describe the interaction between the system and its environment independent of its implementation. The environment includes the user and any other system that interacts with the system. Non-functional requirements describe aspects of the systems that are not directly related to the functional behavior of the system. It includes a broad variety of requirements that apply to many different aspects of the system, such as usability, reliability, performance (i.e. response time, throughput, accuracy, availability etc), and supportability. We also call these requirements as quality requirements. Hence, concerns can be classified as functional concerns and non-functional concerns. Still, we are not interested to propose a new concern elicitation technique because in literature many such techniques are available. Hence, we will explore use cases from the already existing technique of concern elicitation such as viewpoints, use cases, and goals.

6.2 Manage concerns

There are some technical problems which the developers and stakeholders face when modelling a system as use cases. Use case modelling by itself, however, does not constitute requirements elicitation. Even after a use case modeller expert is included, developers still need to elicit requirements from the users and come to an agreement with the client. In this phase of process model, we describe the methods for eliciting information from users and negotiating an agreement with the client. This activity is divided into two sub-tasks: negotiating specification with clients and maintaining traceability. During negotiating specification with clients, all stakeholders including developers and an expert in requirements management present their viewpoints, listen to other viewpoints, negotiate, and come to a mutually acceptable solution. The outcome is joint application design (JAD) document, which is an agreement among all stakeholders and developers. Hence, minimizing requirements changes later in the development process. The ability to follow life of requirements in whole software development lifecycle is termed as traceability. In other words, it gives answers to questions like: from where a requirement comes i.e. who originates it, how it affects other aspects of the system, which components realize the requirement, which test cases checks its realization etc. It enables developers to show that the system is complete, testers to show that the system conforms to its expectations, designers to record the rationale behind the system, and maintainers to determine impact of change. It is very difficult to implement traceability in requirements management. But, it is essential to accommodate changes. Organizing large numbers of requirements into meaningful and more manageable groups and negotiating specification with clients can make traceability easier to implement and maintain.

6.3 Specify concerns

During this phase, all identified concerns are specified using a template, which contains information like name of the concern, description of its proposed behaviour, its source of origin, its classification as functional or non-functional, its possible stakeholders, its responsibilities, contribution list containing a list of concerns that contribute or affect this concern, a list of concerns needed or requested by the concern, and priorities to the concern for possible stakeholders.

6.4 Identify crosscutting concerns

It is significant to identify crosscutting concerns i.e. candidate aspect at early stages because they may create differing situations and result as undesirable affect on later stages of software development. This is achieved by considering the list of required concerns field in concern specification template and building a table to relate concerns to each other and identifying their crosscutting nature.

6.5 Compose concerns

The last phase in our process model is composing concerns. This can be achieved by constructing a table to identify all possible matchpoints. A matchpoint describes which concerns either crosscutting or non-crosscutting should be weaved so that complete integration should be achieved. Further, there should be some conflicting situations due to negatively contribution of two or more concerns on each other. This can be detected using concerns specification template's contribution field where the concerns with same priority contribute negatively on each other and also be weaved on same matchpoint. And finally, composition rules are defined to accomplish this task.

7. CONCLUSION AND ONGOING WORK

This paper has briefly discussed the efforts contributed towards AORE by many research along with their strength and flaws encountered. Some AORE approaches still suffer from the tyranny of dominant decomposition symptom because they are extensions to traditional requirements engineering approaches

and result in scattering and tangling; some are unable to identify and specify both functional and non-functional crosscutting concerns because most of the approaches consider only non-functional concerns as crosscutting; some need to manage concerns and to offer a systematic method to handle conflicts. Thus, the main motivation towards integrated AORE process models is the analysis of existing AORE approaches, incorporating their strengths into a single integrated approach along with new one as concern management. Because, evidence suggests that most of the projects failures are due to requirements. Hence, for avoiding more projects failures, we need to manage requirements and concerns.

In this paper, we have proposed such an approach that incorporates aspect-oriented concepts and which includes concern management as a key separate activity that is not clearly mentioned earlier in literature. It is very difficult to implement traceability in requirements management. But, it is essential to accommodate changes. Organizing large numbers of requirements into meaningful and more manageable groups and negotiating specification with clients can make traceability easier to implement and maintain. The approach supports identification, management, specification, and composition of all concerns.

Our ongoing work is on extending the description of the approach and presenting a detailed integrated AORE process model. Still, we need to validate our approach by applying it on case studies. We are in progress towards it and achieve it in near future. The future work will cover all these aspects.

8. REFERENCES

[1] Pressman, R. *Software Engineering: A Practitioner* '.s *Approach*, 3'd edition, McGraw Hill, 1992.

[2] P. Naur and B. Randell, "*Software Engineering: Report of the Working Conference on Software Engineering*", Garmisch, Germany, October 1968. NATO Science Committee, 1969.

[3] Parnas, D.L. "*On the criteria to be used in decomposing systems into modules*", *Communications of the ACM*, 15(12):1053–1058 [1972].

[4] Dijkstra, E.W. "*A Discipline of Programming",* Prentice Hall PTR, Upper Saddle River, NJ, USA [1997].

[5] Baldwin, C.Y. and Clark, K.B. "*Design Rules: The Power of Modularity Volume 1",* MIT Press, Cambridge, MA, USA [1999].

[6] S. M. Sutton Jr and I. Rouvellou, "*Modeling of Software Concerns in Cosmos*", In Proceeding of the 1st International Conference on Aspect-Oriented Software Development, pg. 127-133, ACM Press, 2002.

[7] D. L. Parnas, "*A Technique for Software Module Specification with Examples*", Communications of the ACM (CACM), 15(5):330–336, 1972.

[8] Elrad T., Filman R., and Bader A. (eds.), "Theme Section on Aspect-Oriented Programming", *CACM*, 44(10), 2001.

[9] Rashid, A., Moreira, A., Araújo, J., "*Modularization and Composition of Aspectual Requirements*", In 2nd Aspect-Oriented Software Development Conference (AOSD'03), Boston, USA, ACM Press. 11-20, 2003.

[10] Baniassad, E., Clements, P., Araújo, J., Moreira, A., Rashid, A., Tekinerdogan, B., "*Discovering Early Aspects*", IEEE Software Special Issue on Aspect-Oriented Programming. 23(1): 61-70, 2006.

[11] Clarke S. and Walker R. J., "Composition Patterns: An Approach to Designing Reusable Aspects", ICSE, 2001.

[12] Suzuki J. and Yamamoto Y., "Extending UML with Aspects: Aspect Support in the Design Phase", ECOOP Workshop on AOP, 1999.

[13] J. Grundy, "*Aspect-Oriented Requirements Engineering for Component-based Software Systems*", IEEE International Symposium on Requirements Engineering, IEEE CS, pp. 84-91, 1999.

[14] Rashid, A., Sawyer, P., Moreira, A., and Araújo, J. "*Early Aspects: a Model for Aspect-Oriented Requirements Engineering*", Proc. of Int. Conference on Requirements Engineering (RE'02), 2002.

[15] E. Baniassad, S. Clarke, "*Theme: An Approach for Aspect-Oriented Analysis and Design*", In Proceedings of the 26th Int. Conf. on Software Engineering (ICSE04), 2004.

[16] Araújo, J. Whittle, and D-K. Kim, "*Modeling And Composing Scenario-Based Requirements With Aspects*" In Proc. of the 12th IEEE International Requirements Engineering Conference (RE 04), 2004.

[17] Jacobson, I.," Aspect-Oriented Software Development with Use Cases", 978-0-321-26888-4, Addison-Wesley, 2004.

[18] A. Moreira, J. Araújo, A. Rashid, "*A Concern-Oriented Requirements Engineering Model*", *Proc. Conference on Advanced Information Systems Engineering*, Portugal, LNCS 3520, pp. 293 – 308, Springer-Verlag Berlin Heidelberg 2005.

[19] Isabel Sofia Brito and Ana Moreira, "*Towards an Integrated Approach for Aspectual Requirements*", 14th IEEE International Requirements Engineering Conference (RE'06), IEEE 2006.

[20] Zhang Jingjun, Li Furong, and Zhang Yang, "*Aspect-Oriented Requirements Modeling*", Proceeding of the 31st IEEE Software Engineering Workshop SEW-31 (SEW'07), Baltimore, MD, USA, 2007.

[21] Moreira, A., Araújo, J., Brito, I., "*Crosscutting Quality Attributes for Requirements Engineering*", In 14th Software Engineering and Knowledge Engineering Conference (SEKE'02), Ischia, Italy, ACM Press. 167 – 174, 2002.

[22] Jing Zhang, Yan Liu, Michael Jiang, and John Strassner, "*An Aspect-Oriented Approach to Handling Crosscutting Concerns in Activity Modeling*", Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I, IMECS 2008, Hong Kong, 19-21 March, 2008.

[23] Busyairah Syd Ali and Zarinah Mohd. Kasirun,"*Developing Tool for Crosscutting Concern Identification using NLP*", IEEE 2008.

[24] G. Mussbacher, "*Aspect-Oriented User Requirements Notation: Aspects in Goal and Scenario Models*", MoDELS 2007 Workshops, LNCS 5002, pp. 305–316, 2008, Springer-Verlag Berlin Heidelberg 2008.

[25] S. Iqbal, and G. Allen, "*Representing Aspects in Design*", presented at 2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering, Tianjin, China , 2009.

[26] S. Hamza and D. Darwish, "*On the Discovery of Crosscutting concerns in Software Requirements*", Proc. Of Sixth International Conference on Information Technology: New Generations, 2009.

[27] Xiaojuan Zheng, Xiaomei Liu, and shulin Liu, "*Use case And Non-functional Scenario Template-Based Approach to Identify Aspects*", Second International Conference on Computer Engineering and Applications, 2010.

[28] I. Jacobson, *Object-Oriented Software Engineering - a Use Case Driven Approach*: Addison-Wesley, 1992.

[29] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", 5th Int'l Symp. on RE, 2001, IEEE CS Press, pp. 249-261.

[30] A. Finkelstein and I. Sommerville, "*The Viewpoints FAQ*", *BCS/IEE Software Engineering Journal*, 11(1), 1996.

[31] The Standish Group. Chaos Report. Technical report, Standish Group International, 1995, http://www.it-cortex.com/Stat_Failure_Rate.htm#The%20Chaos%20Report%20(1995).

[32] The Standish Group, "CHAOS Summary 2009", Technical report, Standish Group International, Boston, Massachusetts, April 23, 2009, http://www1.standishgroup.com/newsroom/chaos_2009.php

[33] L. Rosenhainer, "*The DISCERN Method: Dealing Separately With Crosscutting Concerns*", In Early Aspects, OOPSLA 2005, 2005.