Towards a Domain-Oriented Approach for Identifying Aspects in Software Requirements[*]

Ehab Abdel Nasser
Department of Information Systems
Arab Academy of Science and Technology
Giza, Egypt
ehabx2007@gmail.com

Haitham S. Hamza
Department of Information Technology
Cairo University
Giza, Egypt, 12613
hshamza@acm.org

Abstract

A major challenge in Aspect-Oriented Software Development (AOSD) is the discovery and modeling of aspects in the early phases of requirements and analysis. The quality of the discovered early aspects using most existing techniques is as good as the input provided to these techniques. As such, it is difficult to conclude that one approach or another can effectively identify aspects in a given set of requirements. In this paper, a new approach for discovering early aspects in requirements is proposed. The proposed approach complements existing ones by providing a mechanism to iteratively understand and analyze the problem domain in order to discover relevant and meaningful candidate aspects. The proposed approach is presented and demonstrated through a case study. Results obtained from the proposed approach are compared to those obtained from the well-known Theme/Doc approach. A tool that supports the proposed approach is implemented and outlined as well.

Categories and Subject Descriptors

D.2.1 [**Requirements/Specifications**]: Methodologies—Tools

Keywords

Requirements Engineering, Aspect Oriented Requirements Engineering, Software Stability Model, Formal Concept Analysis

[*]An earlier version of this paper was presented at the Early Aspects 2010 workshop at the 9th International Conference on Aspect-Oriented Software Development. The paper is recommended by Ruzanna Chitchyan and Steffen Zschaler.

Nasser, E. A., and Hamza, H. S. Towards a Domain-Oriented Approach for Identifying Aspects in Software Requirements. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Early Aspects, R.Chitchyan, S. Zsachaler (Eds.), Vol. 2, No. 1 (2010) 50-55

1. Introduction

One of the potential features of a modern software system is to have the ability to cope with the rapidly changing needs of the software domain due to the volatile operational environment where business rules may change, and new extensions may be needed to be added [5, 4]. Accordingly, the software engineering community strives to develop innovative techniques to construct software systems that are maintainable and evolvable, yet simple and resilient. Unfortunately, these properties are not naturally inherited in software systems, and thus, their realization requires careful attention throughout all the phases of software development life-cycle.

Separation of concerns (SoC) is a generic concept that can be used to reduce software complexity by dividing the software into modules such that each module is responsible for different general concerns or features, such as UI presentation, business logic, and data access. Aspect-Oriented Software Development (AOSD) applies the notion of SoC to effectively deal with crosscutting concerns that may induce high complexity when implementing software systems. Concerns of cross-cutting nature may inhibit system evolution if they are not discovered in early development stages [6, 1, 2]. AOSD defines the notion of aspects to deal with crosscutting concerns that interfere with many parts of the software causing high interdependency and tangled representation between logically non-cohesive system components if not properly handled. An aspect is a system construct that can be used to effectively represent functional or non-functional cross-cutting concerns in the system.

Current approaches for discovering early aspects in software requirements depend mainly in the input provided by the analyst without providing specific guidelines on how the inputs to these techniques can be identified from the software requirements and domain. We believe that system concerns must be systematically identified by deeply analyzing the software domain and its requirements. Accordingly, in this paper, we propose a new domain-oriented approach for discovering early aspects in software requirements. The proposed approach provides a systematic mechanism to iteratively understand and analyze the software domain in order to discover relevant and meaningful candidate aspects. The proposed approach is demonstrated by the means of a case study. The resultant list of identified aspects is compared to that obtained by applying the Theme/Doc approach. A tool that supports the use of the proposed approach is described as well.

The rest of the paper is organized as follows. Section 2 provides relevant background and reviews related work. The proposed approach is presented in Section 3. Section 4 demonstrates the use of the proposed approach on a case study. Conclusions are presented in Section 5.

2. Background and Related Work

This section summarizes related work and reviews key concepts used in the proposed approach.

2.1 Aspect-Oriented Requirements Engineering (AORE)

Aspect-Oriented RE was firstly introduced by Grundy motivated by the need for a new perspective that can handle the interactions and relations between components during Component-based system development [6]. Rashid et al. [10] showed that aspects can be vital in early stages during any system development, not just component-based systems. They suggested the first model of *"Early Aspects"* to separate concerns that crosscut several functional and non-functional requirements in the system at the requirement level.

In [7], the *Reusable Aspect Models* (RAM) approach is proposed as a multi-view based modeling approach, where aspect oriented techniques are found to be a potential solution for scalability and consistency challenges as they already successfully help to identify cross cutting concerns and provide means of their composition and interaction.

In [9], an approach for identifying and categorizing concerns using tagging is proposed. Tagging is a flexible technique that is widely used for categorizing any content including text, videos, and images. In this work, tagging was used to guide the business analysis process to find the similarities between discovered concerns by associating them with tags. A tag can be associated to more than one concern and vice versa. In [11], a new architecture description language (ADL) is proposed to support building aspect oriented systems for multi-agent systems (MAS).

The *Theme/Doc* [1] is a requirement analysis approach that uses Theme modeling as a way of representing system features. The Theme model has two types of themes to describe two types of features: base themes and cross-cutting themes. Base themes are those who express certain functionality that do not repeat at different places in the system but they share some behavior with other themes. Cross-cutting themes are those which overlap with many base themes.

The Theme/Doc approach uses a semi-automated process in which developers supply a list of keywords which are the possible system concerns, then a lexical operation is used to identify which requirement statement can be considered as aspectual, pointing to cross-cutting concerns and shows which of the input keywords can be considered as candidate aspects. A graph is generated showing the system concerns, and their interactions and communications linked by requirements statements.

The pure lexical operation may lead to a number of false negative concerns as the user may not be a domain expert in the system domain and he/she may miss important possible concerns. Moreover, this approach may result in false positive concerns as the system cannot understand that some terms in the input list are synonyms to each other.

To improve the performance of the Theme/Doc approach, the Latent Semantic Analysis (LSA) is proposed [8]. The approach enhances the Theme/Doc by finding the relations between text blocks and generates the possible system concerns without the user input list. Despite the improvement brought by the LSA technique; however false negative and false positive concerns were not reduced.

2.2 Formal Concept Analysis (FCA)

Formal Concept Analysis [12] is a mathematical framework that is applied to different domains and it is used to understand the relations between different data sets. FCA contains two main elements: the Formal Context and the Formal Concept.

The formal context is a triple (G, M, I), where G is a set of objects, M is a set of features, I is the binary relation between them. The formal context is represented in a matrix in which each row represents an object, while each column represents a feature. When a certain object contains a certain feature, a mark "X" is inserted in the intersecting cell.

Let O is a subset from the objects set G; $\beta(O)$ is the set of features that are common in all the objects in O. Let F is a subset from the features set M; $\alpha(F)$ is the set of objects where each object contains F. So, formal concept in (G, M, I) is (O, F) such that $\beta(O) = F$ and $\alpha(F) = O$.

Table 1 shows a sample formal context with three objects and four features. As shown in Table 1, list O objects (Obj_1, Obj_2, Obj_3) and each one is attributed with certain features, for example Obj_2 is attributed to features F_2 and F_3.

Table 1: Sample formal context used in FCA

	F_1	F_2	F_3	F_4
Obj_1	X			
Obj_2		X	X	
Obj_3	X			X

2.3 Software Stability Model (SSM)

Software Stability model [3, 5, 4] is a generic modeling approach to derive stable domain conceptual models that require less effort to evolve in response to new and changing requirements. By stable we mean, a model that does not require unnecessarily effort or cost to adapt to new changes. In our approach, the SSM approach is used to systematically analyze the domain and identify its cross-cutting concerns.

SSM partitions the system into three layers [5, 4]:

- EBT (Enduring Business Themes): This is the most abstract description. EBTs are the elements that present the enduring aspects of the underlying business.
- BO (Business Objects): the abstract classes modified to be used in the system. BOs map the EBTs of the system into more concrete objects.

- IO (Industrial Objects): more specialized and customized classes.

Accordingly, the software core will be encapsulated within the EBTs, then BOs. The system external modules which will be subject to changes and modification will be within the IOs. It is worth noting that, the proposed approach exploits the potential of the SSM concepts (See Figure 1) to further analyze input requirements in order to reduce the impact of incomplete and inaccurate requirements.

3. Proposed Approach

Unlike the Theme/Doc approach in which developers supply a set of keywords as candidate concerns, our approach is based on a more concentrate analysis for the software requirements and domain. We believe that such an approach will lead to a more accurate view of the system concerns. Moreover, systematic domain analysis used in the proposed approach can reduce the variation in the quality of the resultant concerns. This is because the quality of the candidate concerns does not rely on the quality of the input provided by the developer as in the case of the Theme/Doc approach.

Figure 1 shows the steps of the proposed approach for identifying system concerns. Given the requirements of the system to be designed, the proposed approach starts with a systematic domain analysis using the concepts of SSM discussed in Section 2.3. The objective of this step is to pinpoint candidate system aspects by identifying EBTs, BOs, and IOs. In particular, an EBT along with its related BOs represent a candidate system concern.

However, the SSM does not guarantee the identification of all possible candidate concerns. This is because not all EBTs can be identified by simply applying the guidelines of the SSM [11, 12]. Several iterations may be needed in order to refine and enhance the quality of the identified concerns. This iterative behavior is reflected in Figure 1 as a dashed line. Further, we capitalize on the fact that BOs and IOs are usually easily identified, as they can be directly extracted from the requirements of the system. In our approach, we use BOs and IOs as means to explore more subtle system concerns by identifying what we call the missing EBTs.

To identify missing EBTs, two steps are performed (See Figure 1): the FCA step and the concept interoperation step. In the FCA step, the relationships among the BOs and IOs with respect to the system requirements are explored. More formally, a formal context is formed in which the BOs and IOs are the objects and system requirements are the attributes. From the generated lattice of this context, all BOs and IOs that share a set of requirements and form a formal concept are considered as a candidate system aspect.

In the Concepts Interpretation step, BOs and IOs are grouped to identify missing EBTs. Each group represents a set of BOs and IOs that frequently co-exist among the identified concepts in the FCA step. The co-existence of a set of BOs and IOs indicates that they contribute together to accomplish a certain system feature. With the help of the software requirements, we can deduce this system feature, which is the missing EBT.

At this point, we have identified all possible EBTs in the system either directly from the SSM model or from inspecting the BOs and IOs using FCA and concept interpretation as discussed above. To this end, each identified EBT represents a system concern, expressed by its related requirements statements. The shared requirements between system concerns will show how these concerns affect each other, and which concern is cross-cutting other concerns in the system.

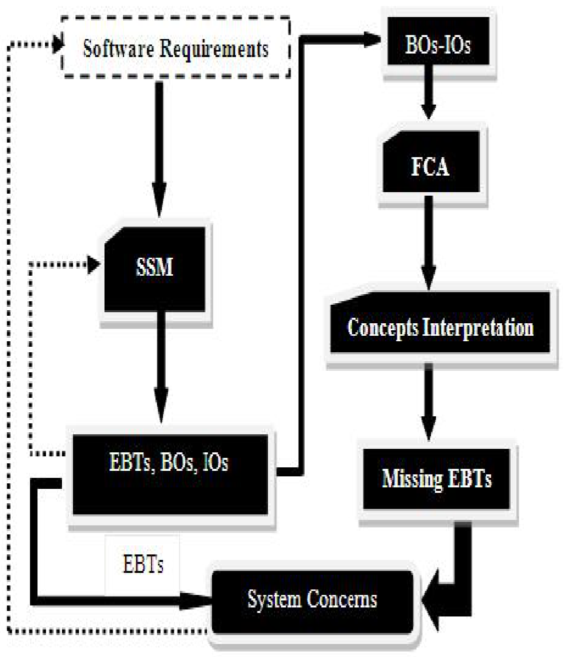
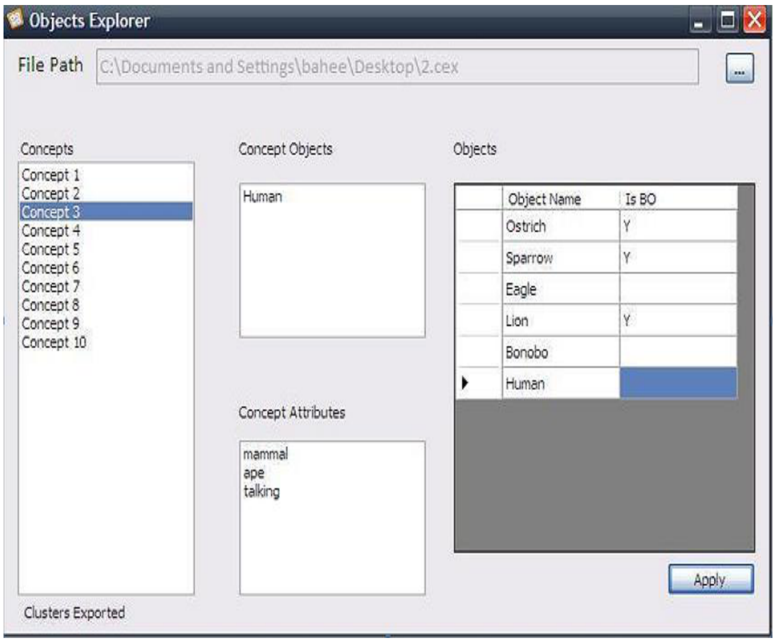


Figure 1: Proposed approach for identifying system concerns

At this stage, we can generate a Theme/Doc view for the resultant concerns similar to the one generated by the Theme/Doc approach. This step will be illustrated in the case study presented in Section 4.

It worth pointing that, in practice, after applying the above process to identify the EBTs of the system, we will have a clear picture for the overall system. Such a clear picture may trigger another cycle of analysis to further refine the input requirements. This point is illustrated as a dashed line in Figure 1 between the System Concerns and Software Requirements.

In order to simplify the implementation of the proposed approach, we developed a tool to semi-automate some of the steps in the approach. In particular, a tool is implemented to perform the steps related to the FCA and concept interpretation. The tool consists of two components: the Concept Explorer and the Object Explorer. The Concept Explorer is a ready-made JAVA-based tool that computes the formal concepts in a given concept and outputs its lattice in XML format. This XML file is input to the Object Explorer in Figure 2 that we implemented from scratch to classify BOs and IOs into groups according to the degree of semantic similarities between these objects. In particular, this process attempts to find how frequent an IO is found in the same concept with a particular BO. For example, IO1 is found with BO1 in 4 concepts, thus these objects have some degree of similar-



File Path: C:\Documents and Settings\bahee\Desktop\2.cex

Concepts: Concept 1, Concept 2, Concept 3, Concept 4, Concept 5, Concept 6, Concept 7, Concept 8, Concept 9, Concept 10

Concept Objects: Human

Concept Attributes: mammal, ape, talking

Object Name	Is BO
Ostrich	Y
Sparrow	Y
Eagle	
Lion	Y
Bonobo	
Human	

Figure 2: Snapshot of the Object Explorer tool

ity and gathered in a group "1". System IO "IO2" is found 3 times with BO1 and 2 times with system BO2, then IO2 is added to group "1" as well, and so on. If an IO is found to have similarity with more than one BO, all such BOs will be added to the same group. The groups generated in this step can help the developer to focus on objects which have semantic similarity and deduce whether they belong to an already existing system concern, or they are pointing to a missing system concern (EBT).

4. Case Study: The Pet Shop

To demonstrate the concepts of the proposed approach, we use it to identify the concerns and aspects in the Pet Shop case study [8]. The Pet Shop system represents the typical requirements of a simple online pet shop. All requirements can be found in [8]. The online pet shop consists of a frontend component (a website for customers to shop and place orders) and a backend component (to process placed orders). The backend component consists of an order fulfillment component to handle orders and ships ordered items, and a supplier component to manage the shop suppliers.

In the following, we present the key steps in applying the proposed approach to the Pet Shop problem statement.

Step 1: Domain Analysis: By applying SSM, we can deduce domain objects and categorize them as follows:

- EBTs: When determining the core EBTs of a system, we have to focus on the services being delivered by the system; we have to be biased towards the customer's point of view. The identified EBTs are: Order Fulfillment, Shipment, Purchasing, and System Administration.
- BOs: They are the instantiations of EBTs, externally stable but not internally in case of system evolution. The identified system BOs are: Order, View, Product, Customer, Transaction, Stock, Front-end Back-end Synchronization.
- IOs: They are explicitly mentioned in the requirements, can be replaced with other alternatives without affecting the system processes, and are not adaptable to system evolution. The identified IOs are: Carte, User Account, Navigation Bar, Search Mechanism, Sign-In Module, Customer Module, Master View, Details View, Shop Carte View, Checkout View, Receipt View, Financial Record, Shopping History Record.

A concern in this step will be defined by an EBT name associated with related requirements. Table 2 shows the identified system concerns (EBTs) from Step 1.

Step2: Concept Identification (Using FCA) and Concepts Interpretation. As previously mentioned in Section 3, the system BOs and IOs will be used as input for FCA to explore their similarities which can help to identify missing system concerns. First, FCA is used to identify concepts and then the concept interpretation step is used to identify missing EBTs.

Table 2: Identified system concerns (EBTs) from Step 1

System Concern	Requirements
Order Fulfillment	R2, R3, R4, R5, R6, R7, R9, R10, R11, R26, R27
Shipment	R5, R6, R7, R8, R14
Purchasing	R3, R6, R13, R22, R26
System Administration	R7, R12, R23, R26

Using the Concept Explorer tool a formal context is created using BOs and IOs as objects and system requirements as requirements. Formal concepts are identified and exported into an XML file as discussed before. Using the Object Explorer tool, the XML file is processed and all objects are grouped according to their semantic similarities. Table 2 gives the output obtained from the Object Explorer process.

- Group 1: the IOs are found with BO ”View” more frequently than with any other BO among the generated concepts, that’s why they gathered in one group. The system BO ”View” has shared many system IOs in the same functionality which is navigating the application products in different view, like Master View, Details View, Shop Carte View, Checkout View, Receipt View, and also giving the user the capability of searching among the shop products—Search Mechanism.
- Group 2: The system BO ”Front-end Back-end Sync” has shared other system BOs and IOs—View, Customer, Order—to accomplish a system feature of creating an order and managing the purchasing process and synchronizing it between the front and back ends to shows a feedback to the application interface about the operation status.

It is worth noting that both system features found in group 1 and 2 focus on the client side—front end—of the system, which was not explicitly explained in details in the requirements. According to the mentioned functional/semantic similarities found between the BO ’View’ and the set o IOs—Master View, Details View, Shop Carte View, Checkout View, Receipt View—supplied by the search capabilities by the BO ’Search Mechanism’, we can identify a missing EBT which is ’System Navigability’. Similarly, functional/semantic similarities found between the system BO ’Async Messaging’ and the other BOs and IOs found in the same group, led us to identify another missing system EBT ’Interface Purchasing Manager’ that is responsible for the purchasing operation and the necessary synchronization between the user interface and the system backend.

Step 3: Theme/Doc View. After identifying the system concerns associated with the system requirements, now we can generate the Theme/Doc view, which helps us to find how concerns affect each other, and which requirements are considered as an aspectual requirement. Figure 3 shows the Theme/Doc view of the proposed approach for the Pet Store case study. To compare the results of our approach we have applied the original Theme/Doc approach. The results are seen in Figure 4. The main concern with the Theme/Doc approach is that it relies on the developer suggested system concerns as an input. This may lead to two main problems: the false negative and the false positive concerns. These two problems are evident from the output of the Theme/Doc approach (See Figure 4). Regarding false negative concerns, we observe that the Theme/Doc output is missing vital system concerns that are not explicitly mentioned in the requirements, which are responsible for showing how the communication between the front end user interface interactions should be with the system backend. Also there is no indication regarding the different views that should be available to the user to browse the products. In addition, the Theme/Doc output is missing another system concern which is related to the administration functionality. The final output also shows that ”Filling Order” and ”Place Order” are two distinct concerns (See Figure 4), although these two are very close and functionally overlapping and should be added to one module. These problems are avoided in our results due to the systematic iterative nature of the proposed approach. It is worth noting that, enhancing the performance of the Theme/Doc approach using the LSA approach [8] adds more complexity to the approach. In LSA, every word in the requirements is treated as a candidate system concern, and then these concerns are filtered to identify the actual concerns. For typical systems with large requirements, such an approach can be very complex.

Table 3: Object Explorer Output

	BO	IO
Group 1	View	Navigation Bar
	View	Search Mechanism
	View	Sign-In
	View	Master-V
	View	Details-V
	View	Carte-V
	View	Checkout-V
	View	Receipt-V
	View	Financial Record
Group 2	View	Shopping History Record
	Product	Shopping History Record
	Order	Shopping History Record
	Sync	Shopping History Record
	Customer	Shopping History Record
	Transaction	Shopping History Record
	Stock	Shopping History Record

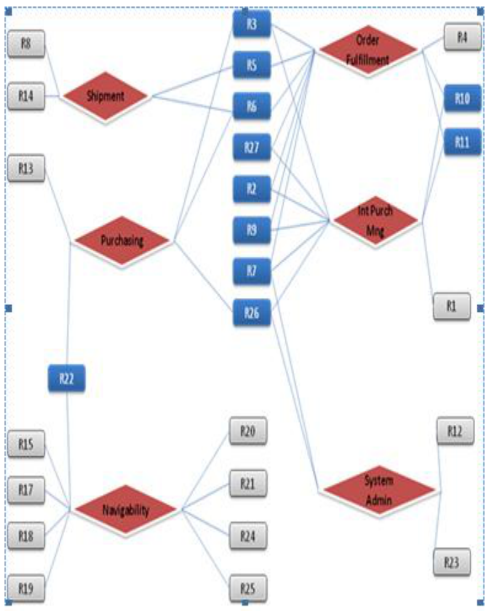


Figure 3: The Theme/Doc view of the proposed approach

5. Conclusions

Representing aspects at early stages of development cycle preserve the homogeneity between developments stages and provide aspects traceability and promoting localization and encapsulation. In this paper, we presented a new domain-oriented approach for identifying candidate aspects in software requirements. The proposed approach is based on systematically identifying system concerns using software stability mode (SSM) and formal concept anal-

Table 4: Missing EBTs identified using FCA

System Concern	Requirements
System Navigability	R15, R17, R18, R19, R20, R21, R22, R24, R25
Shipment	R5, R6, R7, R8, R14
Purchasing	R3, R6, R13, R22, R26
System Administration	R1, R2, R3, R7, R9, R10, R11, R26, R27

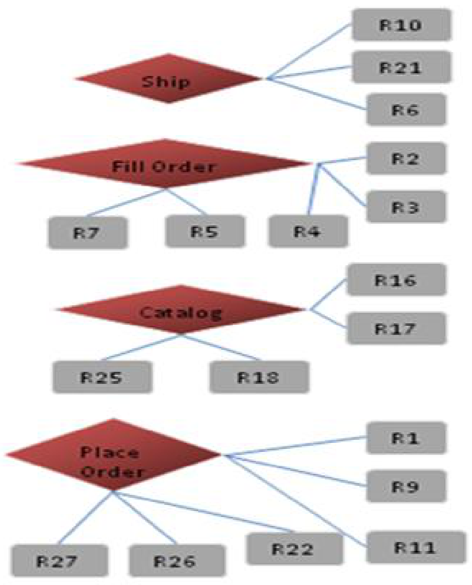


Figure 4: The output of the Theme/Doc approach

ysis (FCA). Identified system concerns are then explored using the Theme/Doc view in order to identify cross-cutting concerns. The proposed approach is applied to the Pet Store case study and the results are compared to those obtained by the Theme/Doc approach. By deeply analyzing the domain of the problem, the proposed approach provided a more comprehensive list of concerns that are difficult to identify using conventional approaches. A tool that supports concern identification and classification is also discussed.

References

[1] E. Baniassad and S. Clarke. Theme: An approach for aspect-oriented analysis and design. Technical report, Department of Computer Science Trinity College, Dublin 2, Ireland, 2004.

[2] R. Chitchyan, M. Pinto, and S. S. Khan. Report on early aspects at icse 2009: Workshop on aspect-oriented requirements engineering and architecture design. *ACM SIGSOFT Software Engineering Notes*, 35, 2009.

[3] M. Cline and M. Girou. Enduring business themes. *Communications of the ACM*, 43, May 2000.

[4] M. Fayad. Accomplishing software stability. *Communications of the ACM*, 45, 2002.

[5] M. E. Fayad, D. S.Hamu, and D. Brugali. Enterprise frameworks characteristics, criteria, and challenges. *Communications of the ACM*, 43, 2000.

[6] J. Grundy. Aspect-oriented requirements engineering for component-based software systems. In *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*. IEEE CS, 1999.

[7] J. Kienzle, W. A. Abed, and J. Klein. Aspect-oriented multi-view modeling. In *Proceedings of the 8th International Conference on Aspect-Oriented Software Development*, Charlottesville, Virginia, USA, Mar. 2009. ACM.

[8] L. K. Kit, C. K. Man, and E. Baniassad. Isolating and relating concerns in requirements using latent semantic analysis. Technical report, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2006.

[9] H. Ossher, D. Amid, A. Anaby-Tavor, R. Bellamy, M. Callery, M. Desmond, J. D. Vries, A. Fisher, S. Krasikov, I. Simmonds, and C. Swart. Using tagging to identify and organize concerns during pre-requirements analysis. In *Early Aspects at ICSE: Workshop on Aspect-Oriented Requirements Engineering and Architecture Design, held with ICSE 2009*, 2009.

[10] A. Rashid, P. Sawyer, A. Moreira, and J. Araujo. Early aspects: a model for aspect-oriented requirements engineering. In *Proceedings of the IEEE Joint International Conference on Requirements Engineering (REŠ02)*, 2002.

[11] C. Silva, J. Castr, M. Lucena, J. Araujo, A. Moreira, and F. Alencar. Support for aspectual modelling to multiagent system architecture. In *Early Aspects at ICSE: Workshop on Aspect-Oriented Requirements Engineering and Architecture Design. held with ICSE 2009*, 2009.

[12] R. Wille. Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications*, 23, 1992.