A Method for Handling Requirements-Level Crosscutting Concerns

Lars Rosenhainer
Department of Computer Science, Chemnitz University of Technology, 09107 Chemnitz
lars.rosenhainer@informatik.tu-chemnitz.de

1 Introduction

Separation of concerns (SoC) is a basic principle of software engineering. It essentially means that dealing successfully with complex problems is only possible by dividing the complexity into sub-problems that can be handled and solved separately from each other. The partial solutions can then be combined to a complete solution. One incarnation of the SoC principle is modularization: each module is responsible for certain concerns of a software system, all modules together realize all the concerns of the software system. In a given problem decomposition, certain concerns may smoothly be encapsulated within modular units. For another sort of concerns, however, this is not possible. They cut across several or even many modular units and are therefore called *crosscutting concerns*. Due to their very nature, crosscutting concerns entail two main problems for software development: scattering and tangling. As to requirements analysis, this means that these concerns are scattered over several or even many statements in requirements documents (scattering), and that one statement often contains information on more than one concern (tangling). However, aspect-oriented software development (AOSD) aims at alleviating these problems by modularizing crosscutting concerns. They are encapsulated by modular units called *aspects* and are thus banned to separate locations. AOSD is therefore an approach to further expanding the SoC principle in software development. With respect to the requirements process, a young subfield of AOSD has been established which is refered to as 'early aspects'.

Handling crosscutting concerns during requirements analysis is critical. It improves requirements traceability, facilitates easier assessment of change impact, supports requirements evolution, and prevents easy overlooking of crosscutting influences [3]. It has for example been shown (see e.g. [5, 3]) that many types of nonfunctional requirements (NFRs), such as response time and security, are crosscutting in nature. They have impact on other requirements which, consequently, cannot be satisfied without taking the former into account. In the next section, a method for dealing with requirements-level crosscutting concerns is described. Afterwards, a conclusion and some remarks on future work are given.

2 The DISCERN method

To address crosscutting concerns separately during the requirements process, the DISCERN method (**D**eal**i**ng **S**eparately with **C**rosscutting Conc**ern**s) [4] is suggested. It is use-case based, iterative and resembles those proposed by e.g. [2, 6] but differs from or extends them at several places. In contrast to most early-aspect approaches, DISCERN explicitly addresses the topic of identifying NFRs in textual documents such as requirements descriptions, as they belong to the most important information sources for system development and evolution. The method steps are described in the following paragraphs.

Identify use cases and actors & specify use cases. As many other methods, DISCERN captures a system's functional requirements by means of actors and their related use cases. Since there are many established methods dealing with the development of use cases, we will not go into detail here. Use cases are specified in the form of scenarios as known from the literature. There is typically one main scenario and a number of alternative scenarios.

Identify NFRs. In many cases, it is trivial to locate statements representing, implying or eventually leading to NFRs in a text. For example, if some statement clearly demands that 'all system responses shall occur within 10 seconds', it is obvious that it represents a response time (RT) requirement. In many other cases, however, it is not trivial. Domain knowledge and much experience as an analyst is necessary to identify NFRs not easily locatable on the text surface. To identify statements related to certain NFR types, DISCERN proposes the application of a reusable and extendable repository of textual patterns. Textual patterns may include e.g. regular expressions for keywords or key phrases (e.g. /immediate/) or certain grammatical structures in combination with keywords (e.g. a noun phrase containing both a cardinal number and a time unit). The usage of textual patterns based on grammatical structures requires a pre-processing step during which certain NLP techniques are applied to a document, which include part-of-speech tagging, sentence boundary disambiguation, lemmatizing, and parsing for deriving syntax trees.

Specify NFRs. If aspects are to be derived from NFRs, DISCERN demands the NFRs to specify unambiguously and in a testable way what constraint or behavior they require and what functional requirements are impacted by them. Interestingly, NFRs of certain types such as security, response time or auditability are typically specified in a way not unsimilar to aspect-orientation since they 'encapsulate' statements as to the *What?* (what is the required constraint or behavior) and the *Where?* (which requirements are impacted). Consider for instance the following RT requirement: 'Orders shall be created within 5 seconds'. It says *what* (within 5 seconds) applies *where* (order creation). Thus it is reasonable that aspect derivation starts with identifying an NFR's *What?* and *Where?* parts. However, there is a difficulty. Since we must deal with natural language, the *Where?* parts are often not specified exactly but may be more or less specific or general. Thus it may be difficult to locate the functional requirements where the *What?* parts apply. Therefore it is desirable to get support in finding additional information on the requirements. Support may for example be offered by searching for statements containing one or more keywords drawn from partial knowledge on an NFR's *Where?* and *What?* parts or similar words by means of thesauri. To this end, DISCERN currently provides for the application of WordNet [1]. Locating relevant information will eventually help in identification and, possibly, necessary refinement of functional requirement descriptions refered to by a *Where?* part. Since DISCERN is use-case based, the functional elements that need to be refered to are scenario steps or chunks of scenario steps.

Specify aspect use cases. An 'aspect use case' is a use case with the particularity of functioning as an aspect. It encapsulates a scenario graph specifying behavior and crosscutting the scenario graphs of one or more non-aspect and/or aspect use cases. It is derived from specified NFRs which are to be operationalized. For describing aspects, DISCERN uses a terminology similar to AspectJ, a Java-based aspect-oriented programming language. A *joinpoint* is an identifiable point in use case specifications. It may be a scenario step or a chunk of consecutive scenario steps with particular properties. A *pointcut* formalizes an NFR's *Where?* part by capturing a set of joinpoints. Finally, a *composition rule* is similar to an advice in AspectJ. It formally defines how a joinpoint is composed with a scenario graph derived from an NFR's *What?* part. The way how elements are composed is controlled by composition rule operators as similarly suggested by [2], which enable the insertion of a scenario graph before or after a joinpoint (*before* and *after*), instead of a joinpoint (*override*) and around a joinpoint (*wrap*).

Compose and validate. Non-aspect and aspect use cases may be composed as defined by the composition rules. Before composition, conflicts must be detected and resolved. DISCERN enables automatic recognition of conflicts between different aspects. Two aspects are in conflict to each other, iff a composition rule of one aspect affects at least one joinpoint whose (a) first step is also the first of a joinpoint affected by a rule of the other aspect and the rules' operators are either *before*, *wrap* or *override*, or (b) whose last step is also the last of a joinpoint affected by a rule of the other aspect and the rules' operators are either *after*, *wrap* or *override*. Conflicts may be resolved by assigning priorities to aspects.

The composed use cases may be validated with stakeholders. Modification to pointcuts and/or composition rules enables alternative compositions, which offer a straightforward way to explore the solution space as to which and how scenario steps are to be affected by NFRs. It may also be determined where there might be conflicts among different NFRs. Further, partial compositions (only including selected aspects) support stakeholder-specific validation as they may describe partial views of a solution proposal, which are relevant to certain stakeholder types (e.g. security expert, end user). Thus diverse compositions help in further refining NFRs.

3 Conclusions and future work

The DISCERN method combines aspect identification from requirements documents with the ensuing creation of aspect use cases and their composition with non-aspect use cases. There are a number of open issues which should be addressed in the future: e.g. extending the textual patterns repository and assessing the effectiveness of textual patterns, handling of NFRs that are not to be operationalized, and mapping of aspect use cases to analysis models such as activity diagrams. All this should of course be accompanied by further evaluation and, possibly, refinement of DISCERN, based on applications to case studies and by improving its tool support.

References

[1] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. The MIT Press, Cambridge, Massachusetts et.al., 1998.

[2] A. Moreira, J. Araújo, and I. Brito. Crosscutting Quality Attributes for Requirements Engineering. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 167–174. ACM Press, 2002.

[3] L. Rosenhainer. Identifying Crosscutting Concerns in Requirements Specifications. In *Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop at OOPSLA 2004, Vancouver, Canada*, 25 Oct. 2004.

[4] L. Rosenhainer. The DISCERN Method: Dealing Separately with Crosscutting Concerns. In *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop at OOPSLA 2005, San Diego, CA, USA*, 16 Oct. 2005.

[5] S. M. Sutton Jr. Concerns in a Requirements Model - A Small Case Study. In *Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop, Boston, MA, USA*, 17 Mar. 2003.

[6] J. Whittle and J. Araujo. Scenario Modelling with Aspects. *IEE Proceedings - Software Special Issue*, 151(4):157–172, 2004.