Workshop on the Modeling and Analysis of Concerns in Software (MACS 2005)

Martin P. Robillard
McGill University, Montréal, Canada
martin@cs.mcgill.ca

Abstract

This report is a summary of the Workshop on the Modeling and Analysis of Concerns in Software (MACS 2005) held at the 27th International Conference on Software Engineering (ICSE 2005). The main goal of the workshop was to bring together researchers and practitioners with interest in techniques for modeling and analyzing the realization of concerns in software systems to support software development and evolution. The workshop consisted of an interactive combination of presentations and discussions. The presentations and discussions were based on a collection of 16 short papers covering a wide range of approaches.

Keywords: Separation of concerns, software modeling, software analysis.

Theme and Goals

Most software design, implementation, and modification activities are organized, explicitly or implicitly, around the idea of concerns. Concerns that arise during software engineering activities typically include features, non-functional requirements, low-level mechanism (e.g., caching), and many other concepts. Programming language-supported constructs like modules, classes, and aspects enable the encapsulation of certain concerns. Unfortunately, because of the limitations of programming languages, structural degradation due to repeated changes, and the continual emergence of new issues, the realization of concerns is often scattered and tangled through the logical decomposition of numerous artifacts (requirements, design diagrams, source code, etc.). Studies and experience have shown that the scattering and tangling of concerns increases the difficulty of evolving software in a correct and cost-effective manner.

The goal of the MACS 2005 workshop was to bring together researchers and practitioners with interest in techniques for modeling and analyzing the realization of concerns in software systems to support software development and evolution, and to explore the potential for integration and interoperability in concern analysis and modeling techniques. Specific themes for the workshop included:

- Concern modeling and representation environments
- Automated and interactive concern location approaches
- Concern mining techniques
- Concern visualization and reverse engineering techniques and tools
- Advanced analysis and design techniques for separation of concerns
- Code transformation and refactoring techniques for separation of concerns

Workshop Organization

We solicited short papers describing ongoing work, new ideas, or recent experience within the scope of the workshop. Each submission was reviewed by the organizers and by members of the program committee. We selected papers for presentation at the workshop based on relevance to the workshop themes and potential to generate interesting discussions. In total we selected three papers for extended presentations and 13 papers for short interactive presentations. The abstract of each paper appears in the printed edition of Software Engineering Notes, and the full text of each paper appears in the on-line edition.

Organization Committee

Martin Robillard, McGill University, Canada
Peri Tarr, IBM T.J. Watson Research Center, USA

Program Committee

Siobhán Clarke, Trinity College, Ireland
Yvonne Coady, University of Victoria, Canada
David Coppit, The College of William and Mary, USA
William Griswold, University of California, San Diego, USA
Rainer Koschke, University of Bremen, Germany
Juri Memmert, JPM Design, Germany
Gail Murphy, University of British Columbia, Canada
Harold Ossher, IBM T.J. Watson Research Center, USA
Arie van Deursen, CWI and Delft University, The Netherlands

Workshop Web Site

http://www.cs.mcgill.ca/~martin/macs2005

Summary of the Presentations[1]

The workshop was divided in four sessions. Each session was intended to explore a specific theme and compare and contrast different approaches associated with or exemplifying the theme.

Concern Modeling

The first session grouped the presentation of different concern modeling approaches and focused on exploring the costs and benefits of modeling concerns at different levels of abstractions.

Just-In-Time Concern Modeling. Martin P. Robillard and Gail C. Murphy. After the workshop introduction, Martin Robillard briefly described the concept of just-in-time concern modeling and its support in the FEAT tool. Just-in-time concern modeling refers to the idea of modeling concerns only when developers first encounter them as part of a program modification task.

[1] Some of the text in this section is edited from material (talks, papers, notes) provided by the workshop participants. Significant portions are indicated in quotes.

Concern Modeling in the Concern Manipulation Environment. William Harrison, Harold Ossher, Stanley Sutton Jr., Peri Tarr. Stan Sutton presented a brief tour of the Concern Modeling Environment (CME) and of the different perspectives on concern modeling offered by CME.

A Model of Software Plans. Robert R. Painter, David Coppit. In the first extended presentation of the workshop, David Coppit presented the idea of mitigating scattered concerns through *software plans*. Software plans are an "approach for dealing with fine-grained concern tangling." The editor-based approach supports a document model that explicitly represents concerns in source code and dependencies between blocks of code.

Mapping Concern Space to Software Architecture: A Connector-Based Approach. Jing Liu, Robyn R. Lutz, Jeffrey Thompson. Jing Liu described an approach to model concerns in the architectural design, and illustrated the idea with a case study of a cardiac defibrillator product line.

Separating Architectural Concerns to Ease Program Understanding. Vladimir Jakobac, Nenad Medvidovic, Alexander Egyed. Alex Egyed described and illustrated concern modeling at the architectural level using the ARTISAn program understanding framework.

Visualization and Transformation

The second session grouped presentations on the topics of concern visualization and concern modeling for the purpose of program transformation. The goal of the session was to explore the trade-offs involved in obtaining intended modularity through visualization versus transformation.

ActiveAspect: Presenting Crosscutting Structure. Wesley Coelho, Gail C. Murphy. In the second extended presentation of the workshop, Wesley Coelho described a technique for presenting crosscutting structure and a tool, ActiveAspect, that supports the technique. With ActiveAspect, "crosscutting structure can be presented in a diagram view without excessive complexity using a combination of user interaction and automated abstraction techniques."

Concern Management for Constructing Model Compilers. Naoyasu Ubayashi, Tetsuo Tamai. Naoyasu Ubayashi described AspectM, a modeling language designed for the management of modeling-level aspects. AspectM improves the separation of concerns pertaining to model transformations in model-driven architecture-based development.

A Model-Driven Approach to Enforce Crosscutting Assertion Checking. Jing Zhang, Jeff Gray, Yuehua Lin. Jeff Gray outlined a "two-level aspect weaving approach to enforce contracts over different abstraction levels." The approach illustrated some of the challenges of weaving assertion-checking code into high-level domain models.

Pattern Transformation for Two-Dimensional Separation of Concerns. Xiaoqing Wu, Barrett R. Bryant, Jeff Gray, Marjan Mernik. Xiaoqing Wu described an approach for two-dimensional separation of concerns allowing developers to transform code back and forth between an object-oriented implementation of the Inheritance design pattern and an aspect-oriented implementation of the Visitor design pattern. This approach allows "the same software to be evolved along different dimensions", enabling developers to choose the most appropriate dimension for a given task.

Concern Mining

The third session focused on the presentation and discussion of approaches to elicit or locate concerns in existing artifacts.

Using Language Clues to Discover Crosscutting Concerns. David Shepherd, Tom Tourwé, Lori Pollock. For the third extended presentation of the workshop, David Shepherd described an approach to automatically identify scattered concerns based on an analysis of the natural language clues found in source code. The proposed approach uses the technique of lexical chaining to locate sets of semantically related terms and their location in source code. Such sets have the potential to denote scattered concerns.

Locating Crosscutting Concerns in the Formal Specification of Distributed Reactive Systems. José J. Pazos-Arias, Jorge García-Duque, Martín López-Nores, Bélén Barragáns-Martínez. José Pazos-Arias presented an approach for the "semi-automatic identification of crosscutting concerns at the requirements level." The approach is intended to improve the incremental process of producing specifications for a distributed system.

Separation of Concerns in Software Product Line Engineering. Mazen Saleh and Hassan Gomaa. Mazen Saleh described an approach and tool supporting the separation of concerns in the context of software product lines. The approach supports "automatic customization of target applications" based on a feature model expressed through a feature description language.

An Exploration of How Comments are Used for Marking Related Code Fragments. Annie T.T. Ying, James L. Wright, Steven Abrams. Annie Ying reported on an empirical study of comments in source code. The main observation of the study is that "programmers mark related code fragments by comments" that either reference related code explicitly or that are similar to other comments located near related code. The conclusion of the study supports the intuition that comments contain valuable clues that can be used to identify and understand scattered concerns.

Concern Analysis

The final session of the workshop focused on concern analysis techniques.

An Approach to Aspect Refactoring Based on Crosscutting Concern Types. Marius Marin, Leon Moonen, Arie van Deursen. Marius Marin "argued for the importance of organizing generic crosscutting concerns by distinctive properties". He proposed the idea of categorizing crosscutting concerns into types. This approach has the potential to help reason about applicable refactorings than can applied to the crosscutting concerns to improve their modularity.

Separation of Concerns for Evolving Systems: A Stability-Driven Approach. Haitham S. Hamza. Haitham Hamza proposed an analysis for early separation and modeling of concerns intended to maximize the stability of a design to reduce the likelihood that scattered concerns will emerge. The approach relies on software stability models and formal concept analysis.

Concern Patterns and Analysis. Juri Memmert. Juri Memmert showed that patterns can be found in the realization of concerns across different software engineering artifacts produced in different phases of the software life-cycle, and how such patterns can be indicative of problems, such as ripple change effects, that will impact the development process.

Summary of the Discussions

In each session the interactions centered on the clarification and discussion of the different approaches presented. In the final session a general discussion took place where the participants focused on defining the term "concern" in the context of software engineering and on synthesizing the workshop.

Defining Concerns for Software Engineering

The participants first agreed that the notion of a concern in software engineering is a very general one and that it changes based on the context in which concerns are considered. Nevertheless, a consensus was rapidly reached in support of the observation that there are two perspectives to the notion of a concern. First, a concern is a conceptual area of interest or focus for a stakeholder of a software project (e.g., a developer). This definition is necessarily vague as it pertains to notions in people's mind, which may be approximate and incomplete. In the second perspective, the term "concern" also refers to the concrete manifestation of conceptual concerns (e.g., in source code, design diagrams, or other artifacts). After converging on a definition of a concern as a conceptual area of interest and its manifestation in a software project, the discussion focused on the issue of mapping or representing a concern's manifestation. In other words, how does one reliably associate a conceptual concern with its concrete manifestation? Many approaches can be used to achieve this goal, including a number of approaches presented at the workshop. While there is currently no consensus on the best way to meet this goal, there was nevertheless agreement on the importance of the question for the purpose of modeling and analyzing concerns in software.

Synthesis of the Workshop

During the first session the different presentations illustrated the need for techniques to help developers model concerns at different levels of abstraction, from source code to architectural design. One opinion expressed by a number of participants is that concerns in the code should be reflected in higher-level artifacts. This observation opened a discussion on traceability and on the potential for concern modeling techniques to facilitate traceability between different artifacts of the software development process.

The presentations of the second session led to a brief debate on the respective goals of concern visualization and transformation techniques. Simply put, the question is as follows: given a concern whose realization is scattered, should we use a visualization technique to view and modify it as a single entity, or should we physically transform the code to encapsulate the concern in a single module? There is no clear answer, and the general consensus was that visualization and transformation approaches are complementary. For example, a visualization approach can be used to understand transformed code, and a transformation approach can be used to refactor a concern into its own module based on an analysis performed with a visualization technique.

The third session focused on concern mining techniques. With two presentations on the analysis of comments and identifiers found in source code, the participants realized the importance of natural language clues for the automatic detection of scattered concerns. As a corollary, the importance of heuristics for automated concern mining approaches was also noted. One open issue emerging from the discussions in the third session is how to bridge the gap between the approximate concern models produced by automated or interactive concern location techniques and the formal models used for code transformation.

The final session on concern analysis emphasized the need to clearly understand the nature of scattered concerns and the potential benefits that can be derived from concern models, including early diagnosis of structural problems in a software system and support for refactoring.

Conclusion

The workshop brought together over 25 researchers and practitioners with interest in techniques for modeling and analyzing the realization of concerns in software systems to support software development and evolution. Sixteen presentations described ongoing work covering an impressive range of approaches that span multiple phases of the software life-cycle (requirements, design, and implementation), and operate at different levels of abstraction (from architectural design to source code). In addition to the ideas, techniques, and tools described, many of the presentations reported on case studies illustrating the different types of concerns that arise during the life of a software project. Taken together, the projects presented and discussed at the workshop form a window on an active field of research at the intersection of many of the traditional sub-disciplines of software engineering. The workshop resulted in a better understanding of the area of concern modeling and analysis and of the open issues in this area. The specific outcome of the workshop includes a definition of the term "concern" in the context of software engineering, 16 papers describing promising concern modeling and analysis techniques, and a record of open issues for the research community.