Analysis and Transformation of Idiomatic Crosscutting Concerns in Legacy Software Systems *

Magiel Bruntink
Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands
and Delft University of Technology
Delft, The Netherlands
Magiel.Bruntink@cwi.nl

Abstract

Legacy software systems often suffer from code quality problems. Maintenance of legacy systems can therefore be costly, and the value of legacy systems may diminish due to a lack of adaptability and reliability. Many code quality problems are caused by idiomatic implementation of crosscutting concerns. This work studies the idiomatic implementation of crosscutting concerns in legacy systems, and examines whether modern language technology like aspect-oriented programming can improve the situation.

1. Introduction

Software systems are bound to suffer from a phenomenon known as *crosscutting concerns* [7]. Crosscutting concerns are concerns –pieces of functionality– that cannot be fit neatly into a system's design. As a result, code scattering, and tangling can occur. An example of a commonly occurring crosscutting concern is error (or exception) handling. Typically, error handling code is *scattered* across the system because each module must deal with exceptional situations. Furthermore, exception handling code tends to be *tangled* with normal (i.e., non-exceptional) code because normal- and exceptional control flow are often correlated.

The problems caused by crosscutting concerns can be observed in legacy software systems in particular, either because legacy systems cannot be easily changed due to their design, or because old programming languages lack the features required (for instance, the lack of exception handling in C). Programmers are then forced to use an *idiom* for the implementation of a concern at every applicable location in the source code. An *idiom* is a prescribed coding solution to a commonly recurring problem. Idiomatic implementation then consists of a manual coding step, in which a programmer produces code by applying the idiom to a particular case.

Hypothesis

The hypothesis of this work is that idiomatic implementation of crosscutting concerns is a source of quality problems in legacy systems. Consequently, idiomatic implementations should be replaced by suitable language technology. Testing of the hypothesis will proceed by means of the following research questions:

1. Has idioms-based implementation of crosscutting concerns led to source code quality problems?
2. Is it possible to improve idioms-based implementations in legacy systems (semi-)automatically? To what extent can this process be automated?
3. What modern language technologies are suitable to replace idiomatic implementations?

2. Motivation

Improving the implementation of legacy software systems may help to consolidate their value, and reduce their cost. By improving the implementation of crosscutting concerns in particular, legacy systems may become more adaptable and reliable, thus more valuable, and more maintainable, thus less costly. Whether it is possible to automate the process of improving legacy systems is not evident. Since the scale of legacy systems is typically such that automation is highly desirable, this question is worthy of study.

Technologies like AOP, DSLs, and modern programming language features (e.g. exception handling) theoretically provide better solutions in a forward engineering setting. Whether these solutions can also be used to improve legacy software systems is not clear in many cases.

*This work is being carried out as part of the Ideals project under the auspices of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Senter program. Thanks to Arie van Deursen for commenting on drafts of this paper.

3. Approach and Results

The research questions are approached within the context of a large-scale industrial C system. This system is being operated, maintained, and developed by ASML, a supplier of lithography solutions in Veldhoven, the Netherlands. The system is responsible for the operation of wafer scanner machines. It consists of approximately 15 million of lines of C code. Crosscutting concerns are prevalent throughout the system, and current implementations of those concerns follow an idiomatic approach.

Within the scope set for the work lie a number of technical problems that have to be solved. The following are unsolved problems in their own right, and are being investigated by the research community.

Identification of (idiomatic) crosscutting concerns. The source code of a system can contain symptoms of the presence of crosscutting concerns. *Aspect mining* or *concern mining* is a subfield of AOSD research in which the automation of the identification of crosscutting concerns is studied. In [5] we report on our study of clone detection in the context of the identification of crosscutting concern code.

Migration of idiomatic crosscutting concerns to modern language technology. Given the results of the identification phase, the question arises how to migrate idiomatic implementations to modern language technology. We discuss the general approach taken in our work in [2]. A verification or checking tool turns out to be key during migrations, and in [4] we show how we developed and used such a tool for the error handling concern.

Analysis and transformation of C code. The C programming language resists easy analysis and transformation in a number of ways, which have been partly solved by the research community. Special technology is required to deal with parsing (context sensitive) C constructs, coping with preprocessing directives, and performing non-trivial analysis. In our work we focused on using the many existing tools for C analysis and transformation. Specifically, in [1] we discuss the leveraging of source code for the purpose of transferring results between tools.

Design of suitable language technology for re-engineering of idiomatic crosscutting concerns. Given an idiomatic crosscutting concern, it is not obvious which language is best suited for its re-implementation. The concepts that need to be present in the target language can possibly be extracted from the current implementation. In [3] we show how concept analysis can help this process. However, even for the simple tracing concern we considered in that work, we observed significant *variability* in the legacy code. We discuss how concept lattices can be used both to identify language features that capture the variability, and to find anomalies that require fixing by a developer.

4. Methods and Evaluation

The work is carried out as part of *Ideals*, a research project that closely collaborates with ASML (see Section 3). Idiomatic crosscutting concerns are studied at different levels within the project, ranging from modeling to source code levels. All research within the project is performed in the industry-as-laboratory style that was proposed by Potts [6]. This style of research entails close integration of research and industry goals and ways-of-working.

5. Expected Contributions

- Evidence of code quality problems caused by idiomatic implementation of crosscutting concerns.
- Several industrial studies that investigate the migration of idiomatic crosscutting concerns to modern language technology like AOP.
- A methodology and tool support for the migration of idiomatic crosscutting concerns.

References

[1] M. Bruntink. Linking analysis and transformations tools with source-based mappings. In *Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM)*. IEEE Computer Society Press, 2006.

[2] M. Bruntink, A. van Deursen, and T. Tourwé. Isolating idiomatic crosscutting concerns. In *Proceedings of the International Conference on Software Maintenance (ICSM'05)*. IEEE Computer Society, 2005.

[3] M. Bruntink, A. van Deursen, M. D'Hondt, and T. Tourwé. Simple crosscutting concerns are not so simple – analysing variability in large-scale idioms-based implementations. In *Proceedings of the Sixth International Conference on Aspect-Oriented Software Development (AOSD'07)*. ACM Press, March 2007. To appear.

[4] M. Bruntink, A. van Deursen, and T. Tourwé. Discovering faults in idiom-based exception handling. In *Proceedings of the International Conference on Software Engineering (ICSE'06)*. ACM Press, May 2006.

[5] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwé. On the use of clone detection for identifying crosscutting concern code. *IEEE Transactions on Software Engineering*, 31(10):804–818, 2005.

[6] C. Potts. Software-engineering research revisited. *IEEE Software*, 10(5):19–28, 1993.

[7] P. Tarr, H. Ossher, W. Harrison, and S. M. J. Sutton. N degrees of separation: multi-dimensional separation of concerns. In *Proceedings of the 21st International Conference on Software engineering (ICSE'99)*, pages 107–119. IEEE Computer Society Press, May 1999.