From Interaction Patterns to Aspects: a Mechanism for Systematic Runtime Monitoring

Ingolf Krüger, Massimiliano Menarini
University of California, San Diego
9500 Gilman Drive, Mail Code 0404
La Jolla, CA 92093-0404, USA
{ikrueger, mmenarini}@ucsd.edu

Michael Meisinger
Technische Universität München
Institut für Informatik
Boltzmannstr. 3, 85748 Garching, Germany
meisinge@in.tum.de

Extended Abstract

Runtime monitoring of software systems requires the insertion into runnable software of monitors that gather information on system states and their evolution. A simple approach to run time monitoring consists of modifying the software source code to implement monitoring functionalities. The very nature of such monitors, however, makes such modifications repetitive and scattered across the whole code base. Aspect oriented languages have been introduced exactly to address repetitive code changes crosscutting the code. They enable a compact representation of such code modifications. Thus, using aspects to specify such monitors seams a promising avenue.

We focus our research on distributed, loosely coupled systems. This system class is based on a well defined communication infrastructure with systems' functionalities emerging from interactions between components over this communication infrastructure. The runtime verification of such systems requires, therefore, monitoring the communications between components and verifying that the expected communication patterns are observed.

We have developed a specification technique for distributed systems based on the use of Message Sequence Charts (MSC) to capture the interaction patterns between entities. Our models are based on a thorough formal foundation and allow for consistent refinement and refactoring. The MSC graphs can, therefore, be used to capture the temporal properties of the *interaction interfaces* of the distributed system. In particular, it is possible to use the models for verification purposes by generating state machine representing the communication behavior of each node of the system. We have applied this strategy, for instance, for conformance testing of components by runtime monitoring [1], generation of executable prototypes for efficient evaluation of multiple architecture candidates ([4], [3]), and to support product-line engineering [5].

When analyzing the relationships between our interaction specification technique and aspect-oriented programming languages ([3], [4], [5]), we were able to identify many similarities between our interaction specifications and aspects. Aspects promote the compact representation of functionalities that spread across different parts of a system's source code. Similarly, our MSC descriptions compactly capture the interaction of logically or physically distributed entities in the system collaborating to provide some functionality. Then, aspects map crosscutting concerns to elements in the program code (pointcuts), whereas MSCs [3] map crosscutting *interaction* concerns to nodes in the distributed system.

These observations motivate the use of aspect-oriented languages as implementation technique for our interaction based models. Thus, we have developed M2Aspects, a code generation tool leveraging the AspectJ language and producing executable system simulations. M2Aspects translates interactions into aspects and uses weaving techniques [2] to establish the mapping between one abstract interaction specifications and low level deployment models. Aspect-orientation propagates the separation of cross-cutting concerns into aspects, maintaining a one to one mapping between models and code.

We propose to combine our interaction specification approach with aspect-oriented technologies to enable an easy modification of distributed systems implementations. We can then embed system monitors into the executable based on models, thus increasing software dependability. We

are investigating enhancements to our M2Aspects tool, resulting in the creation of run-time monitors out of MSC based interaction descriptions. The monitors leverage AspectJ to directly modify an existing Java implementation. We can compactly specify a monitor observing the protocols implemented by the system and insert it into the existing code using the AspectJ weaver without the need of complex code refactorings.

One difficulty is to match aspect language pointcuts, based on code patterns, with the phases of the protocol implemented by the code; this is, in particular, true for systems developed without run time monitoring in mind. We are exploring extensions of M2Aspects that leverage pointcuts that are generally easy to identify at the code level: message send and receive. We leverage our capability to convert interaction patterns to state machines and weave them into the system to keep explicit track of the protocol state. Those state machines can then be used to establish the right pointcuts where the code for monitoring, verifying or even modifying the interactions can be inserted. The generation of the automata, weaved into the system, is based on our algorithm to transform MSCs into state machines, presented in [7].

The use of interaction based specifications has emerged as a powerful abstraction to describe a vast set of real systems. In particular it has been identified as a distinguishing element of service oriented specifications. The notion of *service* has attracted increasing attention both in industry and academia as a mechanism to achieve coupling to address integration of large distributed systems. Service-oriented techniques have been successfully applied in ultra-large scale (ULS) systems. Examples of ULS systems include avionics, automotive, command and control, as well as telematics and public safety systems, to name just a few. In all these domains, the primary challenge to software and systems engineering is the integration of a wide variety of subsystems, their associated applications, data models and sources, as well as the corresponding processes, into a high quality system of systems under tight time-to-market, budget, security, policy, governance and other cross-cutting constraints. These requirements characteristics have led to a high demand for loosely-coupled integration architectures [6]. Therefore, the use of interaction based specifications as a starting point for runtime verification of systems has tremendous potential for increasing quality of real industrial applications.

More work is needed to have a complete and general translation to Aspects implemented in the M2Aspects tool. Moreover, experiments are currently in progress to establish the practicality and usability of this approach in practical applications. Finally, we are investigating the integration of runtime verification, based on the outlined technique, with a full service oriented development process for fail safe systems. We expect this integration to allow us to specify and integrate into existing systems, failure management code to increase the reliability of distributed systems without incurring in the risks introduced by substantial refactoring.

References

[1] J. Ahluwalia, I. Krüger, M. Meisinger, W. Phillips. Model-Based Run-Time Monitoring of End-to-End Deadlines. In Proc. of the Conference on Embedded Systems Software (EMSOFT 2005), 2005.

[2] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect Oriented Programming. Technical report, Xerox Corporation, 1997

[3] I. Krüger, G. Lee, M. Meisinger. Automating Software Architecture Exploration with M2Aspects. In Proc. of the ICSE 2006 Workshop on Scenarios and State Machines (SCESM'06) ACM Press, 2006.

[4] I. Krüger, R. Mathew, M. Meisinger. Efficient Exploration of Service-Oriented Architectures Using Aspects. In Proc. of the 28th Intl Conference on Software Engineering (ICSE 2006), ACM Press, 2006.

[5] I. Krüger, R. Mathew, M. Meisinger. From Scenarios to Aspects: Exploring Product Lines. In Proc. of the ICSE 2005 Workshop on Scenarios and State Machines (SCESM'05), ACM Press, 2005.

[6] I. Krueger, M. Meisinger, M. Menarini, S. Pasco. Rapid Systems of Systems Integration – Combining an Architecture-Centric Approach with Enterprise Service Bus Infrastructure. In Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2006), 2006.

[7] I. Krüger, R. Grosu, P. Scholz, M. Broy: From MSCs to Statecharts, in: Franz J. Rammig (ed.): Distributed and Parallel Embedded Systems, Kluwer Academic Publishers, 1999