Model-Driven Load and Performance Test Engineering in DynaMod*

Eike Schulz[2], Wolfgang Goerigk[1], Wilhelm Hasselbring[2], André van Hoorn[3], Holger Knoche[1]

[1] b+m Informatik AG, D-24109 Melsdorf

[2] Software Engineering Group, Kiel University, D-24098 Kiel

[3] Reliable Software Systems Group, University of Stuttgart, D-70569 Stuttgart

Abstract

Defining representative workloads, involving workload intensity and service calls within user sessions, is a core requirement for meaningful performance testing. This paper presents the approach for obtaining representative workload models from production systems that has been developed in the DynaMod project for model-driven software modernization.

1 Introduction

Workload generation is essential to systematically evaluate performance properties of software systems under controlled conditions. For example, for load, stress, and regression testing or benchmarking it is necessary to expose the system to workload, i.e., to generate requests to provided services. Manual load generation is not practical for various reasons. Hence, automatic generation of synthetic workload is a common practice in performance evaluation [1, 2, 4] and different approaches have been proposed (e.g., [3, 4, 5, 9]). Established tools for generating requests from workload specifications exist, typically based on recorded traces or analysis models. However, one of the biggest challenges is still to obtain *representative* workload specifications similar to production usage.

In the software modernization context, the existing legacy system can be utilized for valid test case generation. An adequate domain (business) model of the legacy system is very useful—if not even necessary—for modernization. Together with usage profiles from production systems, it can be exploited for model-based test development as well.

This paper presents a systematic semi-automatic model-driven approach to obtain and execute representative workload specifications for session-based systems, based on use case specifications and refined by quantitative workload information, such as transition probabilities, think times, behavior mix, etc. The approach was developed as part of our DynaMod project for model-driven software modernization [8], which is depicted in Figure 1. In DynaMod, a combination of static and dynamic reverse engineering techniques is used to reconstruct architectural models of the legacy system. Employing model-driven techniques [7], these models are transformed into architectural models and executable artifacts of the modernized system. Along with the modernized system artifacts, performance tests are generated.

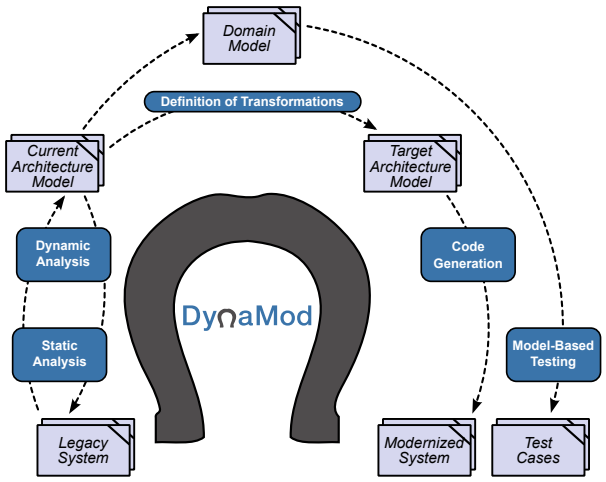

Figure 1: DynaMod software modernization approach

The latter activity forms the scope of this paper, and is summarized in Section 2. Section 3 summarizes the application of the approach to one of the DynaMod case study systems. Section 4 draws the conclusions and outlines future work. A more detailed description of the approach can be found in [6].

2 Workload Model Construction

The workload modeling formalism of our previous work [9] introduces *workload models* consisting of (i) an *application model* and (ii) a weighted set of *user behavior models*, which provides a probabilistic representation of user sessions. A user session is a sequence of related actions by the same user [5].

The workload model construction comprises four complementary steps, as depicted in Figure 2: (i) manual specification of the use cases and (ii) the application model, (iii) dynamic analysis of the production system (PS), and (iv) automatic extraction of user behavior models from the production data. The

*This work was funded by the German Federal Ministry of Education and Research under grant number 01IS10051.

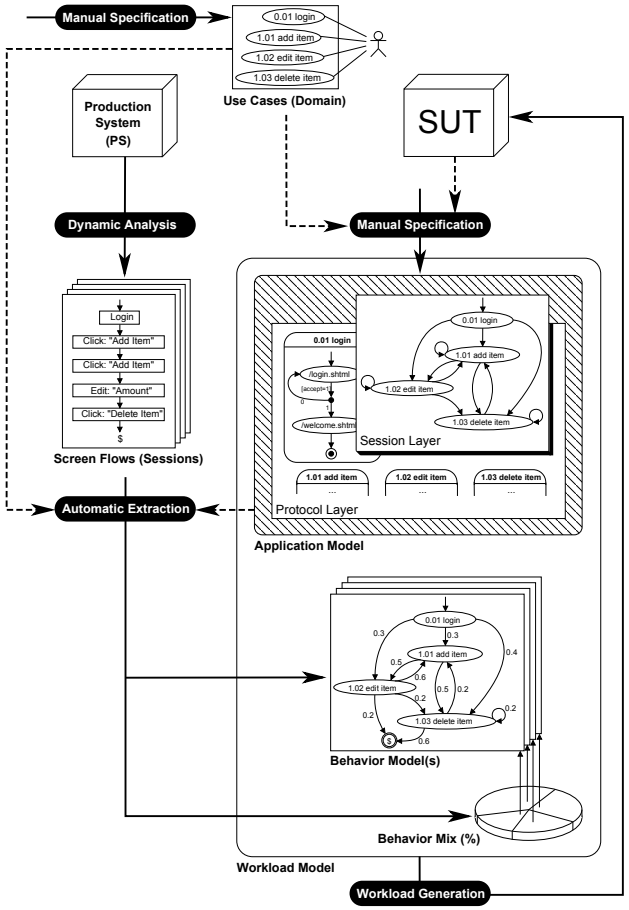

Figure 2: Overview of our approach

behavior and application models are used in a subsequent step to generate test plans to be executed by a suitable load driver.

A basic assumption in our approach is that the production system and the system under test (SUT) share a common domain model (cf. Figure 1), including the set of use cases that are specified manually. The application model is manually created based on the use cases, reflected on the application model's session layer. SUT-specific details are modeled on the protocol layer. Based on a dynamic analysis of the production system, user sessions are reconstructed. The set of user behavior models and its weighting function (behavior mix) are extracted from these sessions. The structure of the behavior models is given by the application model's session layer.

3 Case Study

We applied our approach to one of the DynaMod case study systems [8], namely AIDA-SH, which is an information management and retrieval system for inventory data of historical archives. The system is a client-server application based on the Visual Basic 6 platform and is being used in production by several German archives. As part of the DynaMod research project, a modernized prototype version of the system, called AIDA-Gear, was developed employing model-driven software development techniques [7], which served as the SUT. The set of use cases, from which we created the application model, was provided by domain experts. We obtained runtime traces from the PS instrumented with the Kieker monitoring tool [10]. The extraction of user sessions—in form of executed VB6 UI events—and user behavior models was automated using the tool support for our approach developed in the DynaMod project. The generated workload models were executable on the SUT using JMeter/Markov4JMeter [9].

4 Conclusions and Future Work

We sketched our approach to systematic and semi-automatic creation and execution of representative workloads for load tests of session-based systems based on dynamic analysis results of legacy systems. It is based on previous work on DynaMod [8] and on modeling and executing probabilistic and intensity-varying workloads [9]. Details are also provided in [6]. Future work will be to increase the degree of automation, e.g., by integrating the approach into a model-driven software development platform [7] and to assess the validity of the obtained workload models.

References

[1] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proc. SIGMETRICS '98/PERFORMANCE '98*, 1998.

[2] Raj Jain. *The Art of Computer Systems Performance Analysis*. Wiley, New York, 1991.

[3] Diwakar Krishnamurthy, Jerome A. Rolia, and Shikharesh Majumdar. A synthetic workload generation technique for stress testing session-based systems. *IEEE TSE*, 32(11), 2006.

[4] Daniel A. Menascé. Load testing of web sites. In *IEEE Internet Computing*, 2002.

[5] Daniel A. Menascé, Virgilio A. F. Almeida, Rodrigo Fonseca, and Marco A. Mendes. A methodology for workload characterization of e-commerce sites. In *Proc. EC '99*, 1999.

[6] Eike Schulz. A model-driven performance testing approach for session-based software systems, 2013. Student research paper, Kiel University, Kiel, Germany.

[7] Thomas Stahl and Markus Völter. *Model-Driven Software Development – Technology, Engineering, Management*. Wiley & Sons, 2006.

[8] André van Hoorn, Sören Frey, Wolfgang Goerigk, Wilhelm Hasselbring, and Holger Knoche et al. DynaMod project: Dynamic analysis for model-driven software modernization. In *Proc. MDSM '11*, volume 708 of *CEUR Workshop Proc.*, 2011.

[9] André van Hoorn, Matthias Rohr, and Wilhelm Hasselbring. Generating probabilistic and intensity-varying workload for Web-based software systems. In *Proc. SIPEW '08*, 2008.

[10] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proc. ICPE '12*. ACM, 2012.