Automatic Extraction of Session-Based Workload Specifications for Architecture-Level Performance Models

Christian Vögele,[1] André van Hoorn,[2] Helmut Krcmar[3]
[1] fortiss GmbH, 80805 München, Germany
[2] University of Stuttgart, Institute of Software Technology, 70569 Stuttgart, Germany
[3] Technische Universität München, Chair for Information Systems, 85748 Garching, Germany

ABSTRACT

Workload specifications are required in order to accurately evaluate performance properties of session-based application systems. These properties can be evaluated using measurement-based approaches such as load tests and model-based approaches, e.g., based on architecture-level performance models. Workload specifications for both approaches are created separately from each other which may result in different workload characteristics. To overcome this challenge, this paper extends our existing WESSBAS approach which defines a domain-specific language (WESSBAS-DSL) enabling the layered modeling and automatic extraction of workload specifications, as well as the transformation into load test scripts. In this paper, we extend WESSBAS by the capability of transforming WESSBAS-DSL instances into workload specifications of architecture-level performance models. The transformation demonstrates that the WESSBAS-DSL can be used as an intermediate language between system-specific workload specifications on the one side and the generation of required inputs for performance evaluation approaches on the other side. The evaluation using the standard industry benchmark SPECjEnterprise2010 shows that workload characteristics of the simulated workload match the measured workload with high accuracy.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: measurement techniques, modeling techniques

Keywords

Workload Specifications, Session-based Application Systems, Load Tests, Palladio Component Model

1. INTRODUCTION

In order to validate whether non-functional performance requirements like given response times of application systems can be met, measurement- and model-based performance evaluation approaches are applied [15]. Workload specifications are required for both approaches. Workload specifications serve as input for measurement-based approaches in order to generate synthetic workload to the system under test (SUT), i.e., executing a set of consecutive and related customer requests within a session [9, 10]. Additionally, these specifications are taken into account in formalisms for model-based approaches to predict performance properties early in the software development cycle [3, 8, 15].

To ensure that the measured and predicted performance characteristics of the SUT are similar, corresponding workload specifications must be used. However, there is a lack of approaches enabling the common automatic extraction and specification of workloads for both approaches. The extraction and specification of workloads is done separately for each approach and each tool. This results in additional specification and maintenance effort. The reasons for this are, that these approaches are not integrated and that workload specifications are defined on different levels of detail. Measurement-based approaches need detailed system-specific information like protocol data, whereas model-based approaches are specified on a more abstract level.

In response to these challenges, this paper extends our WESSBAS approach[1] [12, 13], originally developed to automatically extract probabilistic workload specifications for load testing session-based application systems. So far, WESSBAS comprises a *(i)* domain-specific language (WESSBAS-DSL), intended to be a system- and tool agnostic intermediate modeling language for workload specifications, and *(ii)* a transformation to load test scripts. The contribution of this paper is the extension of the WESSBAS approach for model-based performance evaluation. Therefore, we propose a transformation of the WESSBAS-DSL into workload specifications of the Palladio Component Model (PCM) [3], representing an architecture-level performance modeling language. Along with the existing WESSBAS-DSL extraction, this transformation can be exploited by model-based approaches. We focus on architecture-level performance models as they allow to model system architecture, execution environment, and workload specification separately from each other [4]. We evaluate the approach using the SPECjEnterprise2010 benchmark. The developed tools, models, and results of this paper are publicly available online.[2]

LT'15, February 01 2015, Austin, TX, USA.

http://dx.doi.org/10.1145/2693182.2693183.

[1] WESSBAS is an acronym for *Workload Extraction and Specification for Session-Based Application Systems*
[2] http://markov4jmeter.sf.net/lt15

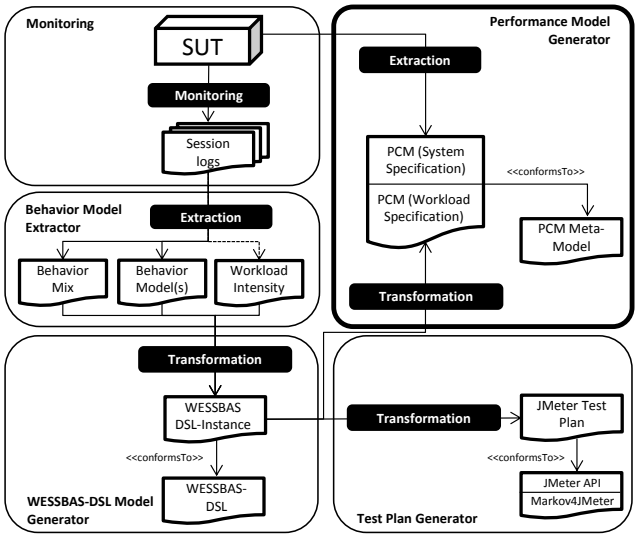

Figure 1: Overview of WESSBAS approach and its extension (bold rectangle), adapted from [13]

2. RELATED WORK

In order to automate the extraction of workload specifications for session-based applications systems, several measurement-based approaches [1, 2, 10] were introduced. These approaches generate workload specifications in tool-specific formats which are not envisaged for model-based approaches.

To evaluate performance properties with architecture-level performance models several approaches were introduced enabling the automatic generation of these models [4, 5]. These approaches focus on the automatic extraction of the system-specific details of the SUT, like the system components, the relationship between the components, the component allocations, and the resource demands. However, the workload specifications must still be modeled manually, which requires a lot of effort for the performance expert.

To reduce the complexity of generating different kinds of analytical performance models from architecture-level performance models several intermediate languages such as PUMA [16] or Klaper [6] were introduced. These approaches only focus on model-based performance evaluations and do not support the definition of workload specifications for session-based software systems.

3. TRANSFORMING WESSBAS-DSL INSTANCES INTO PCM

Before describing the transformation from WESSBAS-DSL instances into PCM workload specifications (Section 3.3), we introduce the required concepts of the WESSBAS approach (Section 3.1) and PCM (Section 3.2).

3.1 WESSBAS Approach

The WESSBAS approach, depicted in Figure 1, introduces a *(i)* domain-specific language (DSL) for layered modeling of workload specifications, *(ii)* an automatic extraction of WESSBAS-DSL instances from session logs, and *(iii)* a transformation from these instances into load test scripts [13]. The performance model generation process, which is the contribution of this paper, is detailed in Section 3.3. The WESSBAS-DSL represents a modeling language for workload specifications of session-based systems. The specification is based on our previous work on the definition and extraction of probabilistic workload specifications [12, 13] for session-based systems. It describes a formalism of workload specifications based on Menascé et al. [10] and Krishnamurthy et al. [9]. The advantage of using this DSL is that it defines the structure of valid workload specifications for session-based systems.

The WESSBAS-DSL enables the modeling of all aspects of a workload model for session-based systems: *Workload Intensity*, *Application Model*, *Behavior Models*, and *Behavior Mix* [12, 13]. The Workload Intensity defines the arrival rates of new sessions as a function over time. The Application Model includes a Session Layer and a Protocol Layer. The Session Layer specifies the allowed sequences of service invocations and SUT-specific details to generate valid requests as extended finite state machine (EFSM). The Protocol Layer models the sequence of protocol-level requests to be executed on the real system. Behavior Models are a probabilistic representation of user sessions in terms of invoked services, associated with Markov States. Transitions between Markov States are labeled with think times and call probabilities. The Behavior Mix specifies the relative frequencies of different Behavior Models representing different customer groups, e.g., Behavior Models for heavy users and/or occasional buyers. These customer groups are identified in the WESSBAS approach using clustering algorithms.

WESSBAS-DSL instances are extracted from recorded session logs of the SUT. From these instances, the test plan generator generates load test scripts for the common load testing tool Apache JMeter[3] including the extension Markov4JMeter [12]. Further details on the WESSBAS approach can be found in [13].

3.2 Palladio Component Model

PCM is a modeling language enabling the prediction of quality-of-service attributes (QoS) like response times, utilization, and throughput [3]. PCM is composed of five complementary model types. The central model type is the *Repository Model*. It models the software components, component operations, and the relations between them. The modeled components are then assembled in a *System Model* to represent the application system. Resource containers (e.g., servers) and their associated hardware resources are modeled in the *Resource Environment Model*, whereas the *Allocation Model* defines the allocation of assembled components to the resource container. The *Usage Model* defines the workload of the system.

As our proposed approach focuses on the generation of PCM workload specifications, the system-specific parts of the model must be created in a separate step. As manual modeling requires too much effort, approaches which automatically extract PCM instance from design specification or running applications (e.g., [4, 5]) can be used to generate the system-specific part of the SUT.

3.3 Transformation

The PCM Usage Model offers only basic support for modeling complex workloads. For example, they grow in complexity for larger workloads due to the lack of reuse con-

[3]http://jmeter.apache.org/

Table 1: Mapping of WESSBAS-DSL concepts to PCM model elements

WESSBAS-DSL	*PCM Model Elements*
Behavior Models	Repository Model (Basic Component, RDSEFF)
Session Layer FSMs	not required
Protocol Layer FSMs	not required
Workload Intensity	Usage Model (Closed Workload)
Behavior Mix	Usage Model (Branch)

cepts. Consequently, we cannot transform the WESSBAS-DSL solely to the usage model. As the Repository Model offers this kind of structuring, we generate parts of the workload specification into the Repository Model (cf. [14]). This violates the clear separation of the PCM models but reduces the complexity of the transformation considerably. Further, this way we do not need to extend the PCM meta-model.

The transformation maps elements of the WESSBAS-DSL to elements of PCM as described in Table 1. First, the existing PCM Repository Model is loaded and for each Behavior Model of the WESSBAS-DSL a component with a corresponding interface used to represent the relationships between the components is generated. For each Markov State of a Behavior Model a component operation, represented as *RDSEFF (Resource Demanding Service Effect Specification)* [3], is created. RDSEFFs describe the behavior of component operations in a way similar to the Unified Modeling Language. Within each RDSEFF, the transitions of the current Markov State to the next states are represented. This way, the allowed sequence of service invocations is controlled by the Markov States themselves. An example can be found in Figure 2. The RDSEFF for the *View_Items* Markov State of the generated Behavior Model component *gen_behavior_model3* has a probability branch with two branch transitions, each representing a transition to the next Markov State. The left and the right transitions have a call probability of 92.9 % and 7.1 % respectively. This branch transition specifies the call probability and contains three different actions:

- First, the think time of this transition is modeled as specified in the WESSBAS-DSL using an *InternalAction* either as mean value or as normal distribution with mean and deviation. In our example, the think time is specified with a mean value of one time unit for both transitions.
- Second, the matching system operation of the modeled SUT is called as an *ExternalCallAction*. This call models a request to the system, e.g., clicking a link of a web page. To identify the corresponding system operation we use a name mapping between the name of the system operation and the name of the Markov State. Only the operations of components providing external system calls will be matched with the Markov State names. In the left transition of our example the operation *View_Items* of the system component with the name *_app* is called as it has the same name as the next Markov State.
- Third, the RDSEFF of this Behavior Model component representing the next Markov State is called as *ExternalCallAction*; in the left transition of our example, the *View_Items* state is called again and in the right transition the state *home* is called. In this way, each Behavior Model component calls itself until a RDSEFF without successor is reached. In this case no further call is modeled and the sequence terminates.

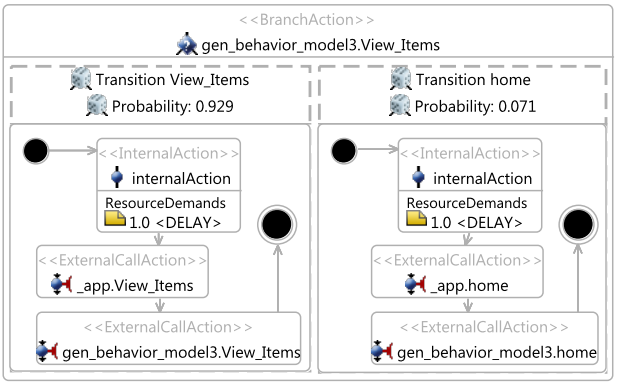

Figure 2: Generated RDSEFF example

Having created the Behavior Model components in the Repository Model, each newly created component is allocated to the System Model and correspondingly to the Allocation Model. A new Usage Model is generated with one probabilistic branch representing the Behavior Mix. For each Behavior Model, a branch transition with the relative frequency as call probability is created. Within this transition the initial Markov State of the Behavior Model is called. Finally, the workload intensity is modeled as closed workload with *(i)* the population representing the number of active sessions and *(ii)* the think time between the end and the start of a new session. The generation of open workloads will be examined in the future.

The Session and the Protocol Layer are not mapped to PCM elements. The Session Layer could be modeled as an additional abstraction layer to the SUT. However, this would increase the complexity of the model and has no impact on the simulation results as the allowed sequences of service invocations are already specified in the representation of the Behavior Models. The Protocol Layer is not used in performance models.

4. EVALUATION

In this section, the practicality and the prediction accuracy of transformed workload specifications will be examined. First, the SPECjEnterprise2010[4] benchmark is briefly explained and then the accuracy of transformed workload specifications is summarized.

SPECjEnterprise2010 represents a Java EE industry application of an automobile manufacturer whose main users are automobile dealers. This benchmark contains a workload specification and a dataset required for load test executions. In this work, we use the Orders domain of the benchmark as the SUT. The Orders domain represents a web-based e-commerce application. SPECjEnterprise2010 defines three different transaction types and in total 13 different HTTP request types. It enables customers purchasing and selling cars (Purchase), managing their accounts and in-

[4]SPECjEnterprise is a trademark of the Standard Performance Evaluation Corp. (SPEC). The SPECjEnterprise2010 results or findings in this publication have not been reviewed or accepted by SPEC, therefore no comparison nor performance inference can be made against any published SPEC result. The official web site for SPECjEnterprise2010 is located at http://www.spec.org/jEnterprise2010/.

Table 2: Evaluation Results

		Orig.	2 Behavior Models		3 Behavior Models		4 Behavior Models	
	Request	MRC	SRC	PE%	SRC	PE%	SRC	PE%
1	add to cart	63,761	64,943	1.82%	61,812	3.15%	60,986	4.55%
2	cancel order	632	609	3.78%	661	4.39%	625	1.12%
3	clear cart	6,047	6,178	2.12%	5,927	2.02%	5,846	3.44%
4	defer order	6,782	6,873	1.32%	6,524	3.95%	6,606	2.66%
5	home	59,934	61,146	1.98%	58,747	2.02%	58,744	2.03%
6	inventory	30,596	30,539	0.19%	29,574	3.46%	29,405	4.05%
7	login	61,500	61,156	0.56%	58,747	4.69%	58,745	4.69%
8	logout	59,934	61,146	1.98%	58,747	2.02%	58,744	2.03%
9	purchase cart	8,360	8,388	0.33%	7,976	4.81%	7,836	6.69%
10	remove	3,027	2,986	1.37%	2,876	5.25%	2,949	2.64%
11	sell inventory	66,679	66,131	0.83%	63,185	5.53%	63,914	4.33%
12	shopping cart	9,074	9,164	0.98%	8,803	3.08%	8,795	3.17%
13	view items	498,601	491,812	1.38%	470,392	6.00%	475,000	4.97%
	$\sum$	874,927	871,071	0.44%	833,971	4.91%	838,195	4.38%

ventory (Manage), and browsing the catalogue of available cars (Browse).

To evaluate the accuracy of the extracted workload specification we employ the evaluation methodology used in our previous paper [13]. The number of simulated requests for the different HTTP request types are compared with the originally measured request counts to the SUT. In our previous work we extracted [13] WESSBAS-DSL instances from session logs of a SPECjEnterprise2010 benchmark run with 800 users, a duration of ten minutes (600 seconds), and the original benchmark Behavior Mix (25 % Purchase, 50 % Browse, and 25 % Manage). Afterwards, we selected three different WESSBAS-DSL instances for that evaluation; one with two, one with three, and one with four Behavior Models. These instance were extracted using different clustering settings; however the resulting workload characteristics are the same [13]. To ensure the comparability with workload specifications of performance models, we evaluate whether the same results can be achieved. We generate a PCM instance representing the SPECjEnterprise2010 system using the approach proposed by [5]. Then, each of the three instances is transformed into this PCM instance. This PCM instance is then simulated for 600 time units corresponding to the ten minutes of the original benchmark run.

The results of the measured request counts (MRC) and simulated request counts (SRC) per HTTP action can be found in Table 2. In addition, for each simulation run the relative prediction error (PE) compared to the measured data is given. The original workload includes 61,500 sessions and in total 874,927 HTTP requests. The relative counts of the request types are very similar for all predicted workloads. Further, the PE of the request types are at maximum 6.69% for purchase cart with four Behavior Models. Thus, from the server-side perspective the SRCs are representative compared to the MRC. As this was the case for load tests extracted from WESSBAS-DSL instances as well [13], the suitability of the WESSBAS-DSL as intermediate language for workload specifications could be demonstrated.

5. CONCLUSION AND FUTURE WORK

Several authors describe the need to integrate measurement and model-based approaches to evaluate the performance of software systems [11, 15]. To close this gap for workload specification, our WESSBAS approach [13] is extended. In this paper, the existing WESSBAS-DSL is used as an intermediate language for the transformation to workload specifications of architecture-level performance models. The evaluation using WESSBAS-DSL instances extracted from the Java EE benchmark SPECjEnterprise2010 demonstrates that representative PCM workload specifications compared to the original workload can be generated. To the best of our knowledge no other approach generates model-based workload specifications for session-based systems from extracted data.

In our future work, we plan to investigate the impact of the extracted workload specification on the measured and on the predicted performance. Furthermore, we plan to evaluate the proposed approach by transforming the WESSBAS-DSL into other architecture-level performance models such as the Descartes Modeling Language [7]. The prioritization of load test cases using the generated performance models will be investigated [14]. Moreover, we plan to implement the transformation between the WESSBAS-DSL and PCM in a bidirectional way. The advantage is when the test cases are analyzed and prioritized within PCM corresponding load test scripts can be generated using the WESSBAS-DSL.

6. REFERENCES

[1] M. F. Arlitt, D. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large web-based shopping system. *ACM TOIT*, 1(1):44–69, 2001.

[2] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proc. SIGMETRICS '98*, pages 151–160, 1998.

[3] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3 – 22, 2009.

[4] F. Brosig, N. Huber, and S. Kounev. Automated extraction of architecture-level performance models of distributed component-based systems. In *Proc. ASE '11*, pages 183–192, 2011.

[5] A. Brunnert, C. Vögele, and H. Krcmar. Automatic performance model generation for java enterprise edition (EE) applications. In *Proc. EPEW '13*, pages 74–88. Springer, 2013.

[6] A. Ciancone, A. Filieri, M. Drago, R. Mirandola, and V. Grassi. Klapersuite: An integrated model-driven environment for reliability and performance analysis of component-based systems. In *Proc. Tools '11*, pages 99–114. Springer, 2011.

[7] S. Kounev, F. Brosig, and N. Huber. The Descartes Modeling Language. Technical report, Department of Computer Science, University of Wuerzburg, 2014.

[8] H. Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 67(8):634–658, 2010.

[9] D. Krishnamurthy, J. A. Rolia, and S. Majumdar. A synthetic workload generation technique for stress testing session-based systems. *IEEE TSE*, 32(11):868–882, 2006.

[10] D. A. Menascé, V. A. F. Almeida, R. Fonseca, and M. A. Mendes. A methodology for workload characterization of e-commerce sites. In *Proc. EC '99*, pages 119–128, 1999.

[11] C. U. Smith. Introduction to software performance engineering: origins and outstanding problems. In *Proc. SFM '07*, pages 395–428, 2007.

[12] A. van Hoorn, M. Rohr, and W. Hasselbring. Generating probabilistic and intensity-varying workload for Web-based software systems. In *Proc. SIPEW '08*, pages 124–143, 2008.

[13] A. van Hoorn, C. Vögele, E. Schulz, W. Hasselbring, and H. Krcmar. Automatic extraction of probabilistic workload specifications for load testing session-based application systems. In *Proc. VALUETOOLS*, 2014.

[14] C. Vögele, A. Brunnert, A. Danciu, D. Tertilt, and H. Krcmar. Using performance models to support load testing in a large soa environment. In *Proc. LT '14*, pages 5–6. ACM, 2014.

[15] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Proc. FOSE '07*, pages 171–187, 2007.

[16] M. Woodside, D. C. Petriu, D. B. Petriu, H. Shen, T. Israr, and J. Merseguer. Performance by unified model analysis (PUMA). In *Proc. WOSP '05*, pages 1–12. ACM, 2005.