Modeling Complex User Behavior with the Palladio Component Model

Christian Vögele,[1] Robert Heinrich,[2] Robert Heilein,[4]
André van Hoorn,[3] Helmut Krcmar[4]
[1] fortiss GmbH, 80805 München, Germany
[2] Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
[3] University of Stuttgart, 70569 Stuttgart, Germany
[4] Technical University of Munich (TUM), 85748 Garching, Germany

ABSTRACT

The specification of workloads is required in order to evaluate performance characteristics of application systems using performance prediction approaches like the Palladio Component Model (PCM). One of the biggest challenges in workload modeling is to ensure that the modeled user behavior adequately resembles the real user behavior. However, PCM offers limited support to model such complex user behavior. Furthermore, reusing modeled activities is not possible. To overcome these limitations, workarounds are required. In order to avoid these workarounds, we extend the meta-model of the PCM Usage Model. We evaluate the extended PCM Usage Model by integrating it into our previous work on automatic extraction of workload specifications. Based on HTTP web logs, recorded from the standard industry benchmark SPECjEnterprise2010, instances of a domain-specific language (DSL) for modeling workload specifications are extracted. Afterwards, these instances are transformed to the extended PCM Usage Model. The evaluation 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

Performance Models, Workload Specification, Palladio Component Model, WESSBAS

1. INTRODUCTION

In order to evaluate the performance of application systems using model-based prediction approaches like the Palladio Component Model (PCM)[1], the modeling of workloads is required. Workloads describe the user behavior and workload intensities in terms of the number of requests to the system under test (SUT) [3]. One of the biggest challenges in workload modeling is, that these models must be representative compared to the real workload [2]. This is especially important to ensure that the predicted performance in terms of response times, resource utilization, and throughput using performance models match the measured performance with high accuracy.

Within PCM, workloads are modeled with the Usage Model. The Usage Model is a domain- specific modeling language allowing to specify workload intensities (i.e., the number of concurrent users), user behavior (i.e., the control flow graph of user system calls), and parameters passed with the system calls [1]. However, modeling complex user scenarios with the Usage Model has limitations, which makes the workload modeling often difficult or even unfeasible. In response to these limitations we extend the Usage Model.

These extensions result in several advantages. First, the modeling of realistic and complex user behavior is possible. Thus, all kind or usage flows extracted from running applications can be modeled without using workarounds like applied in [7]. Second, modeling of business processes (BP) is enabled. BPs are a set of one or more linked activities where each activity itself is composed of one ore more linked steps [9]. Steps are either performed completely by a human actor or performed completely by an information system (IS) [4]. By introducing reusability concepts, BP activities must only be modeled once and can then be reused by other activities.

To summarize, the contribution of our proposed approach comprising the following elements: *(i.)* The extension of the PCM Usage Model, including *(ii.)* the evaluation using the WESSBAS approach [7, 8] against the industry-standard benchmark SPECjEnterprise2010. WESSBAS introduces a domain-specific language (DSL) for modeling workload specifications, an automatic extraction of DSL instances from session logs and a transformation from this DSL into load test scripts and performance models.

2. LIMITATIONS OF PCM USAGE MODEL

The Usage Model meta-model (PCM Version 3.4.1) can be found in Figure 2. The dashed rectangles represent the new elements and are explained in the next section. We use the example of the SPECjEnterprise2010 purchase transaction (see Figure 1) to explain the limitations. SPECjEnterprise2010 is a Java EE industry benchmark representing an application of an automobile manufacturer whose main users are automobile dealers. The Orders domain of this benchmark represents a web-based e-commerce application and enables customers purchasing and selling cars (Purchase), managing their accounts and inventory (Man-

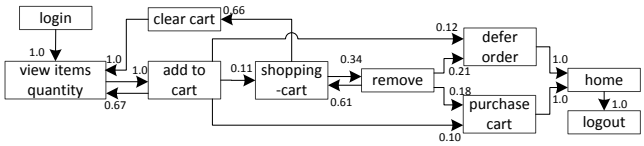

Figure 1: Probabilistic representation of the SPECjEnterprise2010 purchase transaction type

age), and browsing the catalogue of available cars (Browse). Within the purchase transaction, orders are placed and immediately purchased or deferred. The shopping cart is either cleared or items are removed one by one.

The following limitations can be observed. First, an *AbstractUserAction* can only have zero or one successor and zero or one predecessor (see Figure 2). This prevents the modeling of usage behavior, which is representative compared to the real usage behavior. *(i.)* There is no possibility to model backward-edges like from *view items quantity* to *add to cart*. *(ii.) Loops* can be modeled using the *Loop* element. A *Loop* element is a container, within the elements are looped as often as specified. However, when more than one edge leaves the loop, the *Loop* element cannot be used. For instance, the loop *view items quantity*, *add to cart*, *shoppingcart*, *clear cart* cannot be modeled using this *Loop* element, as more than one option is available to leave the loop, e.g., from *add to cart* to *purchase cart* and from *shoppingcart* to *remove*. *(iii.) Branches* are containers as well comprising of multiple *BranchTransitions*. Elements from one *BranchTransition* cannot be linked to elements of another *BranchTransition*. For example, the transitions from the user action *remove* to *purchase cart* or *defer order* cannot be modeled as these actions reside in different branches.

Second, a *UsageScenario* cannot be called by another *UsageScenario*. Within a Usage Model, multiple *UserScenarios* can be modeled. However, there is no possibility that a *UserScenario* calls another *UserScenario* as they are running independent from each other, specifying their own workloads. Therefore, usage flows (c.f. activities) cannot be modeled once and reused by other activities, which is especially important in BP modeling.

Third, only probabilistic *BranchTransitions* can be specified. Thus, the concept of guards and actions (GaA) to control the usage flow cannot be applied. A guard is a condition which must hold true in order to enable a transition. If the transition is executed, an action can change the value of a guard variable [6]. GaA can have an impact on the length of a simulated user session or on the number of simulated requests. For example, a *purchase* action can only be executed when items are added to the *shopping cart* before. Thus, the concept of probabilistic conditions, which is a combination of probabilistic and guarded *BranchTransitions*, must be introduced.

3. EXTENSION OF PCM USAGE MODEL

In this section, the proposed new elements of the PCM Usage Model (dashed rectangles) are explained.

Modeling Complex User Behavior: In order to model representative user behavior the limitation of having only one successor and respectively one predecessor must be overcome. Therefore, we introduce two new *AbstractUserActions*: *MergeBranch* and *DecisionBranch*. A *MergeBranch* has one or many incoming edges and only one outgoing edge. Whereas a *DecisionBranch* has one incoming edge and one or many outgoing edges. Using these new elements, the *AbstractUserActions* *Branch* and *Loop* are not required anymore.

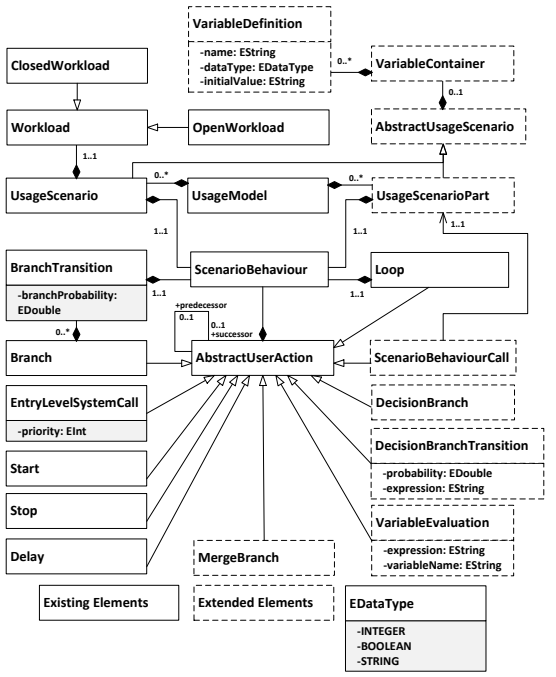

Figure 2: Extended PCM Usage Model meta-model

Reusability: To enable the modeling of reusable activities, we introduce the elements *UsageScenarioPart* and *ScenarioBehaviourCall*. A *UsageModel* can have multiple *UsageScenarioParts* which are similar to *UsageScenarios*, except that they have no workload definition. Both inherit from *AbstractUsageScenario*. Due to the fact that a *UsageScenarioPart* is a reusable scenario it can be called by other *UsageScenarios* and therefore does not need a workload definition. The *UsageScenarioPart* can be called with support of the new element *ScenarioBehaviourCall*. When a *ScenarioBehaviourCall* is called, it executes the linked *UsageScenarioPart* until a *Stop* action is reached. Afterwards, it continues with the superior *UsageScenario*.

Probabilistic Conditions: To model probabilities and GaA, each outgoing edge from the *DecisionBranch* has a new superordinate element called *DecisionBranchTransition*. Within this element a probability must be set. Additionally, a guard condition can be specified. In case the guard is false the edge will be ignored and the probabilities of the other edges are extrapolated to one. Thus, the probabilities are dynamically calculated during runtime. To set an action, the *VariableEvaluation* element must be integrated into the usage flow. The expression will be evaluated and the result is written to the variable defined in the field *variableName*. These variables must be defined in the *VariableContainer*, which is a container for variables used within a *UsageScenario*. Within the *VariableContainer*, multiple variables can be specified with the *VariableDefinition* element. This element defines variables with the attributes *name*, *dataType* (Integer, Boolean or String), and *initialValue*.

Table 1: Evaluation Results

		Orig.	without GaA		with GaA	
	Request	MRC	SRC	PE%	SRC	PE%
1	add to cart	21,376	20,766	2.94%	21,490	0.53%
2	cancel order	342	350	2.29%	285	20.00%
3	clear cart	2,043	2,005	1.90%	2,194	6.88%
4	defer order	2,273	2,237	1.61%	2,249	1.07%
5	home	19,409	19,039	1.94%	19,009	2.10%
6	inventory	19,960	19,452	2.61%	19,609	1.79%
7	login	19,913	19,514	2.04%	19,527	1.98%
8	logout	19,194	18,838	1.89%	18,812	2.03%
9	purchase cart	2,811	2,716	3.50%	2,728	3.04%
10	remove	947	901	5.11%	736	28.67%
11	sell inventory	43,375	42,741	1.48%	42,089	3.06%
12	shopping cart	2,991	2,906	2.92%	2,932	2.01%
13	view items quantity	21,300	20,706	2.87%	21,408	0.50%
14	view items	67,886	66,518	2.06%	65,112	4.26%
	$\sum$	243,820	238,689	2.15%	238,180	2.37%

4. EVALUATION

In this section, the accuracy of the extended Usage Model is evaluated. We first extracted standard HTTP web logs from a running SPECjEnterprise2010[1] deployment. The benchmark run was executed with 800 users, a duration of twelve minutes (720 seconds), three minutes ramp up and ramp down phase, and the original benchmark transaction mix (25 % Purchase, 50 % Browse, and 25 % Manage). Afterwards, we used the WESSBAS approach to generate instances of a domain-specific language (DSL) for modeling workload specifications based on these web logs [7]. Then, we modified the transformation explained in [8] to generate workload specifications using the extended PCM Usage Model. We generated the Usage Model once with and once without GaA.

The accuracy of the extracted workload specification are evaluated by comparing the number of simulated requests for the different HTTP request types with the originally measured request counts to the SUT. The result of the measured request counts (MRC) and simulated request counts (SRC) per HTTP action can be found in Table 1. Further, for each simulation run the relative prediction error (PE) of the SRC compared to the MRC is given.

The evaluation shows that the simulated request counts match the measured request counts with high accuracy. The maximum prediction error without GaA is 5.11% for the request type *remove*. With GaA the prediction errors are slightly higher. The maximum prediction error is again for the request type *remove* with 28.67%. This was expected as GaA do not allow the execution of *remove* when there are no items in the shopping cart anymore.

5. RELATED WORK

We group the related work into approaches for modeling complex user behavior and into approaches for the extraction of workload specifications based on system traces. Due to space limitations we give representative examples.

Modeling complex user behavior: An approach for modeling complex user behavior from a business process perspective is proposed by [4]. The PCM Usage Model meta-model has been extended by user behaviors from a business process perspective. These behaviors can also be none system interactions, like e.g. a user mixing chemicals. In contrast, the proposed meta-model extension in this paper puts focus on modeling complex usage flows.

Extraction of workload specifications based on system traces: The iObserve approach exploits observed method traces for generating the states and transitions of behavioral models and the corresponding usage intensity [5]. Further, another approach for the automatic generation of PCM workload specifications from log files can be found in [8]. Due to the limitations of the Usage Model (see Section 2) large parts of the workload specification are modeled within the PCM Repository Model. The evaluation of this approach showed, that the prediction results match the measured workload with high accuracy. However, the clear separation of concerns of PCM is violated.

6. CONCLUSION AND FUTURE WORK

This paper presents an extension of the PCM Usage Model in order to model complex user behavior. Additionally, we introduce a concept to reuse activities, which is a key requirement for business process modeling. The evaluation using WESSBAS DSL instances extracted from standard HTTP web logs of the Java EE benchmark SPECjEnterprise2010 demonstrates, that PCM workload specifications can be generated, which match the measured workload with high accuracy. As future work, we plan to enable the modeling of asynchronous communications and session abandonments.

7. REFERENCES

[1] 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.

[2] D. G. Feitelson. *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[3] K. Goševa-Popstojanova, A. D. Singh, S. Mazimdar, and F. Li. Empirical characterization of session-based workload and reliability for web servers. *Springer Empirical Software Engineering*, 11(1):71–117, 2006.

[4] R. Heinrich, P. Merkle, J. Henss, and B. Paech. Integrating business process simulation and information system simulation for performance prediction. *Software & Systems Modeling*, pages 1–21, 2015.

[5] R. Heinrich, E. Schmieders, R. Jung, K. Rostami, A. Metzger, W. Hasselbring, R. Reussner, and K. Pohl. Integrating run-time observations and design component models for cloud system analysis. In *9th International Workshop on Models@run.time*, pages 41–46. CEUR Vol-1270, 2014.

[6] M. Shams, D. Krishnamurthy, and B. Far. A model-based approach for testing the performance of web applications. In *Proceedings of the 3rd International Workshop on Software Quality Assurance*, pages 54–61. ACM, 2006.

[7] 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 *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, pages 139–146. ACM, 2014.

[8] C. Vögele, A. van Hoorn, and H. Krcmar. Automatic extraction of session-based workload specifications for architecture-level performance models. In *Proceedings of the 4th International Workshop on Large-Scale Testing (LT)*, pages 5–8. ACM, 2015.

[9] WFMC Terminology. Glossary (WFMC-TC-1011). 1999.

[1] 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.

Acknowledgment: This work has been supported by the Research Group of the Standard Performance Evaluation Corporation (SPEC) and the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution.