Using Performance Models to Support Load Testing in a Large SOA Environment

Christian Vögele, Andreas Brunnert, Alexandru Danciu, Daniel Tertilt
fortiss GmbH
Guerickestr. 25
80805 München, Germany
{voegele,brunnert,danciu,tertilt}@fortiss.org

Helmut Krcmar
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3
85748 Garching, Germany
krcmar@in.tum.de

ABSTRACT

Load testing in large service-oriented architecture (SOA) environments is especially challenging when services are under the control of different teams. It gets even more difficult if services need to be scaled before a load test starts. It is thus important to estimate workloads for services involved in a load test. Service workloads can be specified by the amount of service operation invocations distributed over time. We propose the use of performance models to derive this information for SOA-based applications before executing load tests. In a first step, we use these models to select usage scenarios. Afterwards, these models are transformed in a way that each scenario can be simulated separately from each other. These simulations can predict service workloads for selected usage scenarios and different user counts.

Categories and Subject Descriptors

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

General Terms

Measurement, Performance

Keywords

Load Testing, Service Workload, Usage Scenario, Performance Models, Service-oriented Architecture

1. INTRODUCTION

This paper describes an approach to support load testing in a large SOA environment using performance models. The approach is developed and applied in an ongoing project that transforms an IT landscape into a service-oriented architecture (SOA)[1]. The SOA paradigm describes how loosely coupled software components offer services in a distributed environment [5]. SOA enables the integration of legacy systems and aims at increasing the flexibility of enterprise IT environments. However, the transformation into a SOA is associated with technical challenges. One of the key challenges is to ensure that given performance requirements (i.e. response times for a given workload) are met by enterprise applications in such an environment. Legacy systems that need to be integrated into a SOA are often not designed for this type of interaction. New access patterns and additional software layers lead to different performance characteristics.

Especially for enterprise applications that extensively reuse existing services, it is important to evaluate the performance before they are rolled out to a production environment [5]. However, planning and executing load tests is a cumbersome task. The complexity of this task increases if several teams are involved in this process and the services provided by these teams are not yet in production.

The IT environment in our project context comprises more than twenty complex information systems maintained by several teams in the organization. These information systems provide more than seventy different services in the SOA environment. It is thus important to not only estimate the workload for the enterprise applications but also for the services used during a test. Service workloads can be specified by the amount of service operation invocations distributed over time. Service level agreements (SLA) between service providers and consumers could be violated when the expected service workloads can not be handled by the service deployments in a test environment [4]. Thus, estimating these workloads is necessary so that a test environment can be set up correctly. Otherwise, the performance evaluation results might be interesting for the service providers but not for the enterprise application consuming these services. We propose an approach to support these tasks using performance models.

2. USING PERFORMANCE MODELS TO SUPPORT LOAD TESTING

The applications we are evaluating are intended to be used by 10,000 users in a pilot phase whereas 100,000 users are expected once they are in production. These applications are developed using a model-driven development approach and are modeled as Unified Modeling Language (UML) activity diagrams. The UML activity diagrams contain the application control flow including information about which service operations are called. Additionally, the expected user be-

[1]Project details can be found in our previous work [2]

LT'14, March 22, 2014, Dublin, Ireland.
Copyright 2014 ACM 978-1-4503-2762-6/14/03 ...$15.00.
http://dx.doi.org/10.1145/2577036.2577038.

havior represented as user think times and call probabilities is integrated into the UML activity diagrams. This information is collected by interviewing domain experts.

In order to reduce the modeling effort, UML activity diagrams are automatically transformed into performance models. The Palladio Component Model (PCM) [1] is used as meta-model for the performance models. The PCM modeling notation is closely aligned with the UML notation and is thus easily comprehensible for technical staff in an organization. A detailed description of the transformation process from UML activity diagrams to PCM models can be found in [2].

In the next section, we describe how the PCM models are used to select usage scenarios for load tests. Afterwards, we describe how predictions using these models help to estimate service workloads for selected usage scenarios.

2.1 Selecting Usage Scenarios

A usage scenario is defined as a path from a specified start element to one of the specified end elements of a PCM model. Parameterizable control flow elements like probabilistic branches and loops describe the sequence of the modeled user actions within these paths and represent the information collected from the domain experts [1].

To detect all possible scenarios, the generated PCM models are traversed recursively using a depth-first search. While traversing the PCM models, the call probability for each scenario is calculated by multiplying the probabilities of all branch transitions within a specific path. Users can define thresholds for the minimum likelihood of execution and the minimum (and/or maximum) number of user actions within a scenario. These thresholds help to avoid that too many scenarios are extracted and that endless loops occur.

The result of the depth-first search is a set of usage scenarios including their probability of being called. Additionally, the number of user actions and service operations called during a usage scenario is provided as a result. Based on this information test experts can select a set of scenarios for the load test which match their test goals. Afterwards, load test scripts for each of these scenarios are created manually.

2.2 Predicting Service Workloads

Before executing a load test, the test environment must be scaled according to the expected workload. This is especially important if services involved in a load test need to be scaled in an integration or pre-production environment. Therefore, we use the PCM models to derive predictions about service workloads. Afterwards, the prediction results can be communicated to each service development team. These teams can then ensure that given SLAs can be met for the expected workloads on the services.

However, the PCM models generated based on the UML activity diagrams are modeled in a way that all usage scenarios are represented as one graph. Thus, it is not possible to evaluate only selected usage scenarios because they are not modeled separately from each other. Therefore, we transform the PCM models to represent the usage scenarios independently from each other. Single usage scenarios can now be excluded or included for the following analysis by adjusting their probabilities manually. To exclude single usage scenarios their call probability can be set to zero. The probability of the remaining usage scenarios must then be extrapolated to one.

The PCM models are enhanced with information about worst case response times for service operations as specified in SLAs. Using these transformed models as input for a simulation engine allows to derive predictions for selected usage scenarios. The number of simulated users can be varied to assess the impact of different user counts. The prediction results show how often each service operation is invoked over time. Additionally, the expected throughput for the given usage scenarios can be derived from these results. The simulation time should be chosen according to the planned execution time of the load tests to simplify the analysis.

3. CONCLUSION AND FUTURE WORK

As shown in this paper, predictions based on performance models can simplify the load test planning and execution in a SOA project. The applicability of the approach presented in this work depends on the availability of software models depicting the control flow of enterprise applications. In case other notations are used to represent the control flow, a new transformation from these notations to PCM must be implemented. However, the resulting PCM models can be used for the approach without further adaption.

Future work includes automatic load test script generation for selected usage scenarios. Additionally, we investigate the use of machine learning to prioritize usage scenarios based on the test goals automatically, e.g. using scenarios that are most likely, that include the most service calls or which lead to the highest resource utilization. Capacity planning using performance models enhanced with resource demand information as shown in [3] is another area to pursue in the future.

4. 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] A. Brunnert, A. Danciu, C. Vögele, D. Tertilt, and H. Krcmar. Integrating the palladio-bench into the software development process of a soa project. In *Proceedings of the Symposium on Software Performance: Joint Kieker/Palladio Days*, pages 30–38, 2013.

[3] A. Brunnert, C. Vögele, and H. Krcmar. Automatic performance model generation for java enterprise edition (ee) applications. In M. S. Balsamo, W. J. Knottenbelt, and A. Marin, editors, *Computer Performance Engineering*, volume 8168 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin Heidelberg, 2013.

[4] G. Canfora and M. Di Penta. Service-oriented architectures testing: A survey. In *Software Engineering*, pages 78–105. Springer, 2009.

[5] Y. Liu, I. Gorton, and L. Zhu. Performance prediction of service-oriented applications based on an enterprise service bus. In *International Computer Software and Applications Conference (COMPSAC)*, pages 327–334, Beijing, China, 2007.