Towards Model-based Performance Predictions of SAP Enterprise Applications

Adrian Streitz, Maximilian Barnert, Johannes Rank,
Harald Kienegger and Helmut Krcmar
Technical University of Munich
85748 Garching, Germany
{adrian.streitz, maximilian.barnert, johannes.rank,
harald.kienegger, krcmar}@in.tum.de

Abstract

High-performing Enterprise Applications are the basis for efficient running business processes. In order to evaluate software performance, traditional methods refer to complex test scenarios following the development phase and neglect that problems are easier fixable when discovered early. This paper tackles the problem of late performance evaluations and presents a conceptual approach that enables response time predictions for SAP Enterprise Applications during the development phase. We introduce a performance model generator that transforms ABAP source code into Palladio Component Model instances by using Abstract Syntax Trees which allows to conduct early performance simulations. Our approach supports conditional and probabilistic control flows to improve prediction accuracy. Based on subsequent performance simulations, we predict response time of applications and their underlying processing systems.

1 Introduction

Performance, considering response time, resource consumption and throughput, has become an important key factor for today's application systems [4]. Furthermore, the efficiency of software systems correlates with a company's revenue [9]. Although application costs become more expensive the later software changes are performed, traditional software engineering methods still conduct performance evaluations after the application's development [1].

Tůma (2014) introduced the concept of *Performance Awareness* [6] to counteract this problem. His idea is based on observing software performance during the development phase and providing support to react upon them. However, we have shown in our previous work that most software developers miss dedicated performance skills, which hinders a significant improvement in software performance [10]. To reduce complexity of manually creating test scenarios and long-running evaluations, we propose to generate performance models automatically and conduct simulations to retrieve performance predictions.

Until now, model extraction approaches are mainly focused on the Java domain. Although Enterprise Resource Planning (ERP) software is highly used in industry, it is still neglected for performance considerations. SAP, as the most common representative for ERP software [5], is based on its own programming language ABAP. Due to existing differences between ABAP and Java, it requires to adapt performance modeling for the ABAP development domain.

Our paper presents an approach that transforms ABAP source code automatically into Palladio Component Model (PCM) instances by the intermediate step of using Abstract Syntax Trees (ASTs). Existing specialties of ABAP (e.g. object-oriented and procedural programming paradigm) make it necessary to introduce a new code-to-model-transformation with parametrizations which include basic operations, as well as external calls to existing components.

2 Related Work

There is a long tradition in predicting performance metrics of software applications. They are mainly based on either component measurements or formal descriptions with automata simulations. The latter include performance models such as PCM [3].

Kappler et al. (2008) present a process to generate PCM instances from Java by combining static and dynamic code analysis [2]. The static analysis parses source code into ASTs that represent their structure and control flow. The dynamic analysis is based on behavioral specifications which are derived from executions and analyzed with statistical methods.

Danciu et al. (2015) showed an approach to provide Java EE developers predicted performance metrics during the implementation phase [7]. For this purpose, Java source code is transformed into PCM instances and enables the conduction of static performance analysis. In addition, calls to external components are parameterized with values provided by the Kieker monitoring framework. The approach is integrated into the programmer's IDE and provides

immediate feedback to the developer.

Keller et al. (2016) argue that current approaches neglect software's design and specification phases [8]. They present a software specification tool which addresses the transition issues by employing a Meta Programming System that makes it easier to move perspectives between different life-cycle stages. The approach allows source code annotations of performance requirements and is able to transform the code into PCM instances for simulation purpose.

3 Approach

Based on characteristics of the SAP application server, we created a performance model generation process for SAP Enterprise Applications. Unlike other commercial software, the source code of these systems is accessible and provides context information for further analysis. Additionally, they store and provide runtime measurements of existing applications.

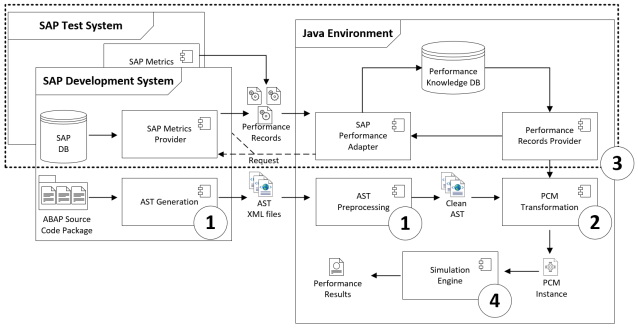


Figure 1: Conceptual Approach

Our developed approach consists of four stages: (1) AST handling, (2) PCM transformation, (3) integration of performance records and (4) model simulation to gain performance estimations. The software architecture and the main steps are depicted in Figure 1.

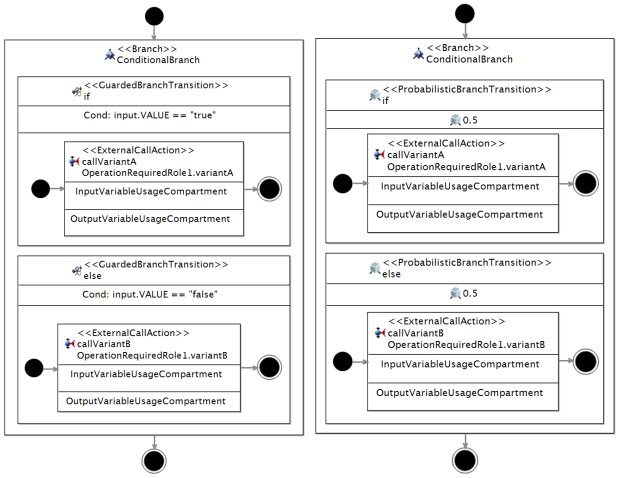
The first step starts with transforming ABAP source code into ASTs. The approach requires a package with one or multiple source files. To tackle software modularization and distributed files, we chose ASTs as intermediate artifact that are able to represent entire applications in one XML structure and support multiple programming languages.

Secondly, the ASTs are parsed and transformed into PCM instances. Since ASTs only provide structural information, we need to parameterize the PCM instance. The *Performance Records Provider* component complements the necessary information by enriching the model with parametrization data.

SAP offers a broad range of built-in tools to measure performance metrics. Our purpose is to make this data externally available. The component *SAP Metrics Provider* receives requests, conducts performance measurements, and exports the gained information. Afterwards, the *SAP Performance Adapter* parses the records and stores them in a separate Performance Knowledge Database. The external database leads to a reduction of workload on the SAP system by providing available performance data directly. Furthermore, it is able to aggregate performance results referring to same operations to avoid statistical outliers.

The final step concludes with a model simulation and results with performance estimations of the inspected application. Basis for the simulation is the enriched PCM instance, which serves as input for the simulation engine.

```
function X(input)
    if input = true then
        VARIANT A()                    ▷ long execution time
    else
        VARIANT B()                    ▷ short execution time
```



(a) Guarded Branch (b) Probabilistic Branch

Figure 2: Conditional Branch Types

One feature of our transformation is to support both conditional and probabilistic if statements. In contrast to other approaches that are solely based on probability distribution, our model allows predictions of dedicated code paths instead of simulating all existing control flows. In certain cases, this leads to more precise predictions if one branch consumes significantly more performance than others do.

Figure 2 shows an example where either variant A (long execution time) or variant B (short execution time) is called depending on the input value of function x. If we are able to evaluate the value of *input*, we transform the provided code into a *guarded* (see Fig. 2a), otherwise a *probabilistic branch* with equal distribution (see Fig. 2b).

By applying our approach, developers do not only get insight about the overall execution behavior, but also how input values influence the resulting performance. Since conditional branches sometime lead to complex expression and static code analysis cannot trace all dynamic values, we propose a mixed transformation approach that uses probabilistic branches as fallback for non-resolvable conditions.

4 Experiment

We chose for our experiment an application that determines ABAP code clones. The application contains procedural, as well as object-oriented instructions. In addition, parts of the entire application are outsourced by several include files. Furthermore, it contains nested loops and branches and uses branch conditions that affect external call actions. The application offers two execution variants (A and B) with different response time behavior to calculate similarity of source files. In order to measure the application's response time behavior, we used the built-in SAP tool *ABAP Code Inspector*.

We setup a virtual machine with 0.5 processing units and 22 gigabytes memory. As application server, we installed SAP NetWeaver 7.4 SP16 on top of an IBM DB2 10.5.5 database and IBM AIX 6.1. Our hardware platform is equipped with an IBM Power E870 server based on 40 cores (4.19 GHz) and 4 terabytes memory.

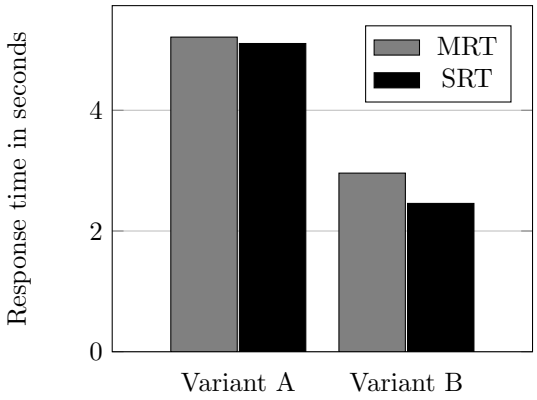


Figure 3: Measured and Simulated Response Times

A bar chart of the measured response time (MRT) and the simulated response time (STR) is depicted in Figure 3. For variant A, the average MRT is 5.21 seconds and the average SRT 5.10. This results in an accuracy of 97.89 %. However, response time predictions of variant B resulted in an accuracy of 83.39 %. Our analysis show that the approach yields accurate performance results if good parametrization metrics are available. The deviations in variant B mainly occur because not all branches could be resolved correctly and were treated probabilistic.

5 Conclusion

The paper fills the missing consideration of model-based performance predictions for ERP software and presents an approach to transform ABAP applications into PCM. With our approach, developers can detect performance issues immediately during the development phase and avoid problems of late software changes. Since prediction and simulation are fully automated, developers neither need to conduct performance evaluations manually, nor require any expertise of the performance engineering domain. In order to retrieve more precise prediction values, the transformation process supports both conditional and probabilistic control flows. Our long-term vision is to extend our approach by considering CPU utilization and the integration into Eclipse IDE to highlight performance bottlenecks in the developed source code directly.

References

[1] B. W. Boehm and P. N. Papaccio. "Understanding and controlling software costs". In: *IEEE Transactions on Software Engineering* 14.10 (1988), pp. 1462–1477.

[2] T. Kappler et al. "Towards Automatic Construction of Reusable Prediction Models for Component-Based Performance Engineering". In: *Software Engineering* 121 (2008), pp. 140–154.

[3] S. Becker, H. Koziolek, and R. Reussner. "The Palladio component model for model-driven performance prediction". In: *Journal of Systems and Software* 82.1 (2009), pp. 3 –22.

[4] A. Brunnert et al. "Performance Management Work". In: *Business & Information Systems Engineering* 6.3 (2014), pp. 177–179.

[5] Computerwoche. *Market share of leading companies in enterprise resource planning (ERP) software in Germany from 2011 to 2013*. 2014.

[6] P. Tůma. "Performance Awareness: Keynote Abstract". In: *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*. ICPE '14. Dublin, Ireland: ACM, 2014, pp. 135–136.

[7] A. Danciu et al. "Performance Awareness in Java EE Development Environments". In: *Proceedings of the 12th European Workshop on Performance Engineering*. EPEW '15. Madrid, Spain: Springer, 2015, pp. 146–160.

[8] F. Keller et al. "Leveraging Palladio for Performance Awareness in the IETS3 Integrated Specification Environment". In: *Softwaretechnik-Trends* 36.4 (2016).

[9] C. Heger et al. "Application Performance Management: State of the Art and Challenges for the Future". In: *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*. ICPE '17. L'Aquila, Italy: ACM, 2017, pp. 429–432.

[10] A. Streitz et al. "Performance Improvement Barriers for SAP Enterprise Applications: An Analysis of Expert Interviews". In: *Proceedings of the 9th ACM/SPEC International Conference on Performance Engineering*. ICPE '18. Berlin, Germany: ACM, 2018, pp. 223–228.