Leveraging Palladio for Performance Awareness in the $\mathrm{IET_S}^3$ Integrated Specification Environment

Fabian Keller,[1] Markus Völter,[2,3] André van Hoorn,[1] Klaus Birken[3]

[1] University of Stuttgart, D-70569 Stuttgart
[2] voelter — Ingenieurbüro für Softwaretechnologie, D-70327 Stuttgart
[3] itemis AG, D-70565 Stuttgart

Abstract

Performance is an important concern when designing and implementing software-intensive systems. Various techniques are available for specifying and evaluating performance concerns throughout the system life-cycle. However, there is a gap in terms of tooling when moving between requirements, design, and implementation artifacts. We address this gap by integrating simulation-based and analytical performance prediction tools into $\mathrm{IET_S}^3$ — an integrated specification environment for technical software systems based on the JetBrains MPS language workbench.

In this paper, we provide an overview of our work in progress on integrating performance awareness support into the $\mathrm{IET_S}^3$ editor and user interface. We leverage Palladio's prediction infrastructure by transforming to Palladio's modeling language to obtain performance predictions, which are then fed back into the $\mathrm{IET_S}^3$ user interface. The approach yields a tight integration of the requirements and the design of a system strengthened by a real-time feedback loop.

1 Introduction

While the performance of a yet to be developed system is hard to assess, it inevitably is an important non-functional requirement to many applications that is often underspecified. Designing systems with performance in mind saves costs in the long run, as architectural changes are often expensive to carry out in an existing system.

Software performance research has developed approaches to predict the performance before the systems are actually built [10]. The predictions are based on the component architecture enriched with performance-specific annotations such as resource consumption and typical use cases. Tooling support is available, but is not deeply integrated with other tools and artifacts produced in the software development process, such as performance requirements or variability models.

Performance awareness captures the (real-time) availability of performance information in system development environments to inform and assist developers [13]. Existing approaches (e.g., [2, 5, 8]) neglect system specification, particularly the design and requirements stages and do not support software product lines (SPLs).

This paper provides a summary of our work in progress on adding performance-awareness support into $\mathrm{IET_S}^3$, an "Integrated Specification Environment for the Specification of Technical Software-systems", based on performance predictions provided by the Palladio [3] tooling environment for model-based performance prediction.

The JetBrains Meta Programming System (MPS) [1] is an open-source language workbench [6], that is, a system for defining, composing and using languages and their IDEs. It supports concrete and abstract syntax, type systems, and transformations, as well as IDE aspects, such as syntax highlighting, code completion, find-usages, diff/merge, refactoring, and debugging. MPS is used widely in industry for languages in a wide variety of domains, including embedded software, insurance, health and medicine, as well as aerospace.

In our $\mathrm{IET_S}^3$ research project, we develop an IDE-like tool for writing specifications of software systems, that seamlessly integrates informal, semi-formal and formal specification languages. It relies on JetBrains MPS and its facilities for language composition and notational flexibility. At the core, the $\mathrm{IET_S}^3$ tool suite comes with four languages: expressions, requirements, components, and feature models to support SPLs. One of the first concrete tools built on top of the $\mathrm{IET_S}^3$ languages is SimBench [4], a tool for discrete performance simulation, which is the foundation for our approach to introduce performance awareness into the tool suite.

2 Related Work

Related work primarily addresses performance awareness for developers during the implementation phase of a project. Horký et al. [8] use performance unit tests to generate performance documentation for the developers using the components to make informed decisions. Danciu et al. [5] employ Palladio to predict the performance of Java EE components during their

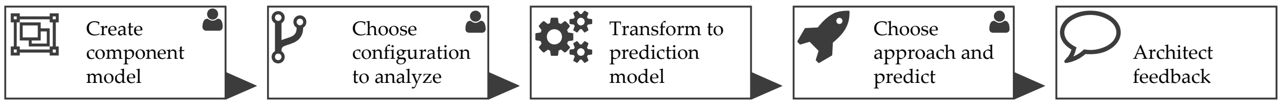

Create component model
Choose configuration to analyze
Transform to prediction model
Choose approach and predict
Architect feedback

Figure 1: Conceptual overview of the approach. A user icon in the top right of a step indicates user interaction.

implementation within an IDE. Performance prediction results are visualized next to the affected source code. Beck at al. [2] integrate performance profiling results directly into the code. Related work also covers awareness with respect to other quality attributes, such as correctness [14]. Our approach integrates performance awareness into a specification environment, such that performance analyses are automatically related to specified requirements.

3 Conceptual Approach

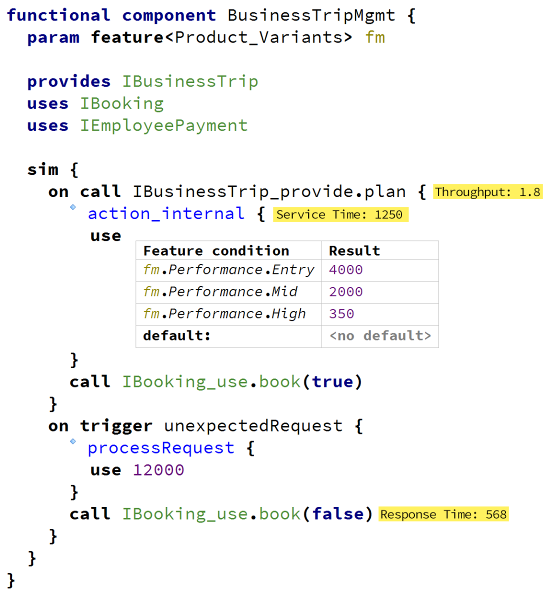
The goal of the approach presented in this paper is to add performance awareness to a specification environment. Hence, the specification environment must treat software performance as first-class citizen for component developers, system architects, and domain experts already during the requirements elicitation and design phase of the software to be built.

IETS^3 provides several language concepts to express component specifications, system architectures built with the components, planned allocations onto hardware, and usage scenarios for the designed system. On top, IETS^3 provides a feature model that tightly integrates with the aforementioned concepts to express variability points on all aspects of the model. The IETS^3 DSL provides all information required by Palladio to conduct a performance prediction for the system, thus a model-to-model transformation of the IETS^3 meta-model to the Palladio component model has been implemented to leverage the performance prediction features Palladio provides. A general overview of the approach is depicted in Figure 1 and consists of the following steps:

1. **Create component model:** The architects and domain experts model the system and its requirements. Component developers supply the components with performance-related annotations to specify the expected performance behavior. Figure 2 shows an example.

2. **Choose configuration to analyze:** If feature modeling is used, the performance prediction can only be performed for a certain configuration, which is either chosen by the user or by heuristics.

3. **Transform to prediction model:** The chosen variant is automatically transformed into an instance of the Palladio component model. The concepts and semantics of the IETS^3 model are retained in PCM, although slight differences in the two modeling languages exist (e.g. processor configuration).



```
functional component BusinessTripMgmt {
  param feature<Product_Variants> fm

  provides IBusinessTrip
  uses IBooking
  uses IEmployeePayment

  sim {
    on call IBusinessTrip_provide.plan { Throughput: 1.8
      action_internal { Service Time: 1250
       use
          | Feature condition       | Result       |
          | fm.Performance.Entry    | 4000         |
          | fm.Performance.Mid      | 2000         |
          | fm.Performance.High     | 350          |
          | default:                | <no default> |
      }
      call IBooking_use.book(true)
    }
    on trigger unexpectedRequest {
      processRequest {
       use 12000
      }
      call IBooking_use.book(false) Response Time: 568
    }
  }
}
```

Figure 2: A sample component in the IETS^3 notation, supporting calls (regular component connections), triggers (performance analysis specific methods) and variant specific resource demands. The yellow boxes show attached analysis results.

4. **Choose and execute prediction approach:** In addition to the discrete event simulation implemented in SimBench, our approach connects to the performance prediction infrastructure provided by Palladio — currently focusing on the Layered Queuing Network solver [11]. We plan to investigate heuristics that help choosing an approach to optimize the end-user experience in terms of the trade-off between prediction precision and analysis runtime [15].

5. **Architect feedback:** The results of the prediction are annotated to the affected parts in the model inside IETS^3. Depending on the analysis, the user may be able to see additional details upon interaction with the prediction result.

4 Implementation

This section highlights selected parts of the current implementation of the approach.

Model-to-Model Transformation: The transformation is built with the MPS Java language, which contains language enhancements tailored to building DSLs with MPS. An open-source builder API[1] to construct PCM models from Java has been built to ease the writing of the transformation.

As the models of IETs_S^3 and Palladio are not semantically identical, a mapping to retain the semantic meaning of IETs_S^3 in the transformed PCM instance was implemented. The sample component in Figure 2 shows a component defining performance analysis relevant metadata under the *sim* keyword. Resource demands of regular component interactions are defined with the *on call* keyword and are directly mapped to Palladio's resource demanding service effect specifications (RDSEFFs). In addition to that, the *trigger* concept allows components to define arbitrary communication and invocation paths regardless of their structure and hierarchy in order to model environmental events and their performance impact (such as the sudden re-routing while playing radio in a car infotainment system). During the transformation the triggers are mapped to artificial PCM interfaces which are connected appropriately in the resulting PCM system. The variable resource demand is evaluated for a specific variant before transforming to the PCM, such that the feature-specific resource demand is a regular numeric value during the transformation.

Performance Prediction: Once transformed, the Palladio component model is passed to the existing solver infrastructure for Palladio, particularly using the simulations and the PCM2LQN transformation [11]. The results of the performance prediction are then lifted to the IETs_S^3 model in order to be shown in the user interface.

As MPS is based on a JetBrains IDE, the Eclipse environment required to run Palladio is not available. To the best of our knowledge, Palladio does currently not provide a headless runner to run analyses without a graphical user interface. Hence, we added the ability to run Palladio analyses without a graphical user interface outside Eclipse environments to our open-source library.

5 Conclusions and Future Work

Our project is an early stage. Hence, we are not able to provide quantitative results at this point in time. One of the next steps is a systematic evaluation w.r.t. performance and scalability using an industry-scale case study. Moreover, we plan to improve the performance evaluation and awareness support for software product lines, leveraging existing techniques from other contexts [7]. To improve the real-time feedback loop we plan to support incremental performance predictions, as users typically only change small parts of the model [12]. Also, a transparent selection of model-based and measurement-based evaluation techniques are of high interest to improve the usability [15].

[1]`http://github.com/DECLARE-Project/palladio-headless`

Acknowledgment

This work is supported by the German Research Foundation (HO 5721/1-1, DECLARE) and the German Federal Ministry of Education and Research (IETs_S^3). More details on this work are provided by Keller [9].

References

[1] Jetbrains MPS. `http://www.jetbrains.com/mps`.

[2] Fabian Beck, Oliver Moseler, Stephan Diehl, and Günter Daniel Rey. In situ understanding of performance bottlenecks through visually augmented code. In *Proc. ICPC '13*, pages 63–72, 2013.

[3] Steffen Becker, Heiko Koziolek, and Ralf H. Reussner. The Palladio component model for model-driven performance prediction. *J. Syst. Software*, 82(1):3–22, 2009.

[4] Klaus Birken, Daniel Hünig, Thomas Rustemeyer, and Ralph Wittmann. Resource analysis of automotive/infotainment systems based on domain-specific models: A real-world example. In *Proc. ISoLA'10, Part II*, pages 424–433, 2010.

[5] Alexandru Danciu, Alexander Chrusciel, Andreas Brunnert, and Helmut Krcmar. Performance awareness in Java EE development environments. In *Proc. EPEW '15*, pages 146–160, 2015.

[6] Sebastian Erdweg, Tijs van der Storm, Markus Völter, Meinte Boersma, Remi Bosman, William R Cook, Albert Gerritsen, Angelo Hulshout, et al. The State of the Art in Language Workbenches. In *Proc. SLE '13*, 2013.

[7] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wasowski. Variability-aware performance prediction: A statistical learning approach. In *Proc. ASE '13*, pages 301–311, 2013.

[8] Vojtech Horký, Peter Libic, Lukás Marek, Antonín Steinhauser, and Petr Tuma. Utilizing performance unit tests to increase performance awareness. In *Proc. ICPE '15*, pages 289–300, 2015.

[9] Fabian Keller. Introducing performance awareness in an integrated specification environment, 2016. Master's thesis, University of Stuttgart.

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

[11] Heiko Koziolek and Ralf H. Reussner. A model transformation from the Palladio Component Model to Layered Queueing Networks. In *Proc. SIPEW '08*, pages 58–78, 2008.

[12] Tamás Szabó, Sebastian Erdweg, and Markus Voelter. IncA: A DSL for the definition of incremental program analyses. In *Proc. ASE '16*, 2016. To appear.

[13] Petr Tůma. Performance awareness. In *Proc. ACM/SPEC ICPE '14*, pages 135–136, 2014.

[14] Markus Voelter, Daniel Ratiu, Bernd Kolb, and Bernhard Schätz. mbeddr: Instantiating a language workbench in the embedded software domain. *Autom. Softw. Eng.*, 20(3):339–390, 2013.

[15] Jürgen Walter, Andre van Hoorn, Heiko Koziolek, Dusan Okanovic, and Samuel Kounev. Asking "what?", automating the "how?": The vision of declarative performance engineering. In *Proc. ACM/SPEC ICPE '16*, March 2016.