# The Palladio-Bench for Modeling and Simulating Software Architectures

Robert Heinrich, Dominik Werle, Heiko Klare, Ralf Reussner, Max Kramer
Karlsruhe Institute of Technology
Karlsruhe, Germany
{heinrich,dominik.werle,heiko.klare, reussner,max.kramer}@kit.edu

Steffen Becker
University of Stuttgart
Stuttgart, Germany
steffen.becker@informatik.uni-stuttgart.de

Jens Happe
avenyou GmbH
Karlsruhe, Germany
jens.happe@avenyou.com

Heiko Koziolek
ABB Corporate Research
Ladenburg, Germany
heiko.koziolek@de.abb.com

Klaus Krogmann
LogMeIn
Karlsruhe, Germany
klaus.krogmann@logmein.com

## ABSTRACT

Software designers often lack an understanding of the effects of design decisions on quality properties of their software. This results in costly and time-consuming trial-and-error testing, delayed and complicated rollouts of the software. In this tool demonstration paper we present an integrated tool environment – the Palladio-Bench – for modeling and analyzing software architectures. The analysis results provided by Palladio support making design decisions by identifying the best-suited design from a set of given alternatives.

The demonstration video for the Palladio-Bench can be found at the URL https://youtu.be/vG7WQPcp-uI.

**ACM Reference Format:**
Robert Heinrich, Dominik Werle, Heiko Klare, Ralf Reussner, Max Kramer, Steffen Becker, Jens Happe, Heiko Koziolek, and Klaus Krogmann. 2018. The Palladio-Bench for Modeling and Simulating Software Architectures. In *ICSE '18 Companion: 40th International Conference on Software Engineering , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3183440.3183474

## 1 INTRODUCTION

For engineers in various engineering disciplines, it is common practice to simulate models before realizing a physical artifact. Design models of cars, electronic devices or buildings are analyzed for understanding the impact of design decisions on quality properties like efficiency, reliability, or safety. Predicting the quality properties of an artifact based on design models without actually having realized it, is essential for identifying adequate designs and making trade-off decisions. In contrast, in software engineering too often engineers lack an understanding of the impact of design decisions on the system's quality properties. Missing understanding of the system's quality leads to costly trial-and-error cycles by partially implementing the software to figure out design flaws.

In this tool demonstration paper we present the Palladio-Bench, an integrated tool environment for modeling and analyzing software architectures. It implements the Palladio approach [9] named after the Italian Renaissance architect Andrea Palladio. Palladio is a system architecture modeling approach for predicting quality properties based on software component design. Initially Palladio was focused on performance and then has been extended for various other quality properties, like reliability [3], scalability and elasticity [5], energy consumption [11], and maintainability [10]. The idea is to avoid costly changes to software after it has been implemented by simulating the quality of software architecture during development. Decisions in software design are typically made on the basis of experiences or, when lacking those, by making an educated guess. Through the information provided by the Palladio approach, the best-suited design alternative can be selected and trade-off analysis can be made.

On the occasion of the releases of the Palladio book [9] and the new version PCM 4.1[1] we present our tool to the broad software engineering community. The new release comprises new extensible graphical editors, new default simulator, new data storage and evaluation framework. Previous tool demonstrations are outdated by the new release [8] or were limited to specific subfeatures [6].

The paper is structured as follows. Sec. 2 describes a running example for illustrating the application of the Palladio-Bench. The essential parts of the Palladio-Bench are described in Sec. 3. The graphical editors and analysis views of the Palladio-Bench are explained in Sec. 4. The paper concludes with a summary and description of future development in Sec. 5.

## 2 RUNNING EXAMPLE

This section introduces a running example – the Media Store system [12] – to demonstrate how Palladio supports understanding the consequences of architectural design decisions. In the running example, we investigate the consequences of design decisions with respect to the quality property performance. Media Store represents

[1]https://sdqweb.ipd.kit.edu/wiki/PCM_4.1

a file-hosting system for uploading and downloading audio files. It provides basic functionality of a file-hosting system like user and media management, encoding and watermarking. An overview of the Media Store's components is given in Fig. 1.

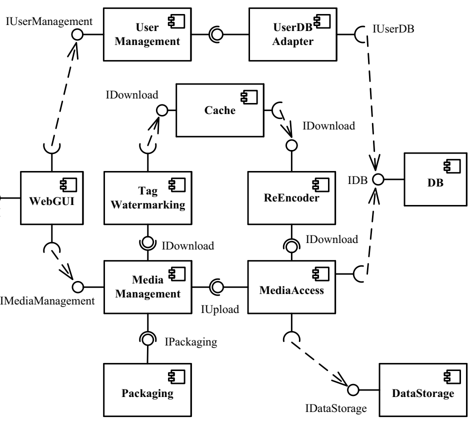


**Figure 1: Architecture of Media Store [9]**

While designing the architecture of the Media Store, a software architect has to be concerned with quality requirements (here performance requirements). In the Media Store system the disk or the CPU might become a bottleneck to resources. In case of the disk, an I/O-intensive component such as the database may be the root cause for the bottleneck, as it reads and writes data most frequently. In case of the CPU, a CPU-intensive component such as the Tag-Watermarking or the ReEncoder component may result in a CPU bottleneck, as watermarking and re-encoding of the audio files cause CPU load. The main questions for the architect are: If the workload increases, which resource would become the bottleneck? When could the system no longer meet the quality requirements?

Additional design decisions like adding a Cache may arise. Although we can generally assume that adding a Cache component would improve the performance of the Media Store by reducing the load on resources like disk and network. However, a logical Cache may lead to increasing load on the server and thus be a bottleneck. Further important questions to address while designing the architecture of the Media Store are: Could the choice of a Cache component decrease the average response time? Would the new Cache component be a potential bottleneck?

For answering questions like the aforementioned, software architects can apply the Palladio approach implemented in the Palladio-Bench to predict the quality properties of component-based software architectures.

## 3 THE PALLADIO-BENCH

The Palladio approach is composed of three essential parts that are designed to work hand in hand. The Palladio Component Model (PCM) as a domain-specific language (DSL) is targeted at specifying and documenting software architectural knowledge. On the basis of the PCM, various analytical techniques ranging from queuing network analysis to discrete event simulation can be applied to predict the quality of the modeled system. The Palladio approach is aligned with a development process comprising several developer roles tailored for component-based software systems. The three parts of the Palladio approach are implemented in the Palladio-Bench which is based on the Eclipse IDE.

The PCM consists of several partial metamodels reflecting different architectural views on a software system as depicted on the left hand side in Fig. 2. The Palladio-Bench provides graphical editors for enabling the different developer roles specifying the partial models of the Palladio approach.

The component repository model is created by the component developer and specifies the software components and their interfaces stored in a repository. Moreover, the component developer describes the components' inner behavior in so-called Service Effect Specifications. For performance analysis in the Media Store the Service Effect Specifications expresses internal actions reflecting the resource consumption of the component's services depending on its context and external service calls.

The software architect specifies the software architecture in the system model by assembling components from the repository. Thus, the performance of the Media Store's architecture can be estimated with respect to the several components in the system model.

The system deployer specifies the processing resources (CPU, hard disk, and network) in the Resource Environment Model. For the initial version of the Media Store we assume only one server. If the system is distributed we need to consider the network and the additional servers. For each server, quality-relevant properties like processing rates of the CPU are part of the Resource Environment Models. Moreover, the system deployer describes the deployment of the components to the resources in the allocation model.

The domain expert specifies the user behavior and usage intensity (i.e. workload) of the system in the Usage Model. For the Media Store we assume an open workload [9]. A user can either register or log into the system, if already registered. The systems shows a list of all available files to the user. The user can download or upload files. Performance-relevant properties like path probabilities and number of files downloaded at once are part of the Usage Model.

For estimating the quality properties of the Media Store design alternatives, the Palladio-Bench offers several quality analysis tools depicted on the right hand side in Fig. 2. The main focus is on performance analysis for which a wide range of tools are available. The Palladio-Bench also offers tools for reliability and cost analysis as well as various extensions, e.g. for maintainability [10] or security [13] analysis. Further quality properties can be considered in Palladio by extending the PCM with quality-specific model elements and providing analysis tools for the quality properties.

An analysis tool in the context of Palladio is a software that implements one or more analysis techniques for resembling the quality properties of a system under study. Analysis tools for estimating the performance of the software system are central to the Palladio approach. The performance tools highlighted in the grey box in Fig. 2 differ mainly in their range of functions, result accuracy and analysis speed.

The deprecated SimuCom [2] is a discrete-event performance simulator which estimates response times of both system-level and component-level services as well as resource utilization. SimuLizar [1] analyzes self-adaptations in cloud-computing environments, e.g. when scaling out components by replication. EventSim [7] is a

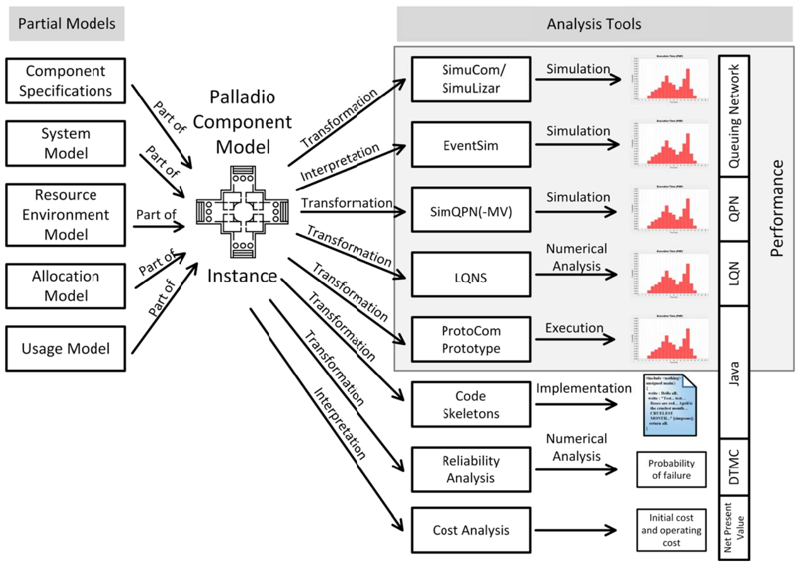


**Figure 2: Partial models of the PCM and transformation to various analysis tools [9]**

discrete-event performance simulator which completes SimuCom in that it primarily addresses highly complex simulation models by applying event-scheduling simulation techniques. Besides the simulation tools, the Palladio-Bench provides tools for transforming Palladio models to the formalisms Queuing Petri Nets (QPN) and Layered Queuing Networks (LQN). These formalisms are independent of the Palladio approach and commonly used for software performance prediction. Finally, ProtoCom [2] is a tool of the Palladio-Bench to create performance prototypes in form of Java code that mimic demands to different types of processing resources to evaluate their performance in a realistic environment.

The Palladio-Bench also provides a reliability analysis tool for estimating software and hardware failure potentials using discrete-time Markov chains (DTMC). A simple cost analysis offered by the Palladio-Bench can be used to assign costs to components and hardware which is then used to estimate the initial and operating costs of the system.

The Palladio-Bench has been successfully applied in several industrial settings including but not limited to the modeling and analysis of a large distributed e-mail system operated by Germany's largest e-mail provider, 1&1 Internet AG, for comparing design alternatives for storage virtualization in IBM systems, and for design exploration of a distributed software system from ABB [9].

## 4 MODEL EDITORS AND ANALYSIS VIEWS

The Palladio-Bench enables developers to create PCM instances with graphical editors (completely reimplemented in the new release) and derive quality metrics from the models using analytical techniques and simulation. Hence, high-level system properties can be tracked down to individual components and even further to component services (e.g., name server, caching, or middleware) and resources (CPU, hard disk, or network).

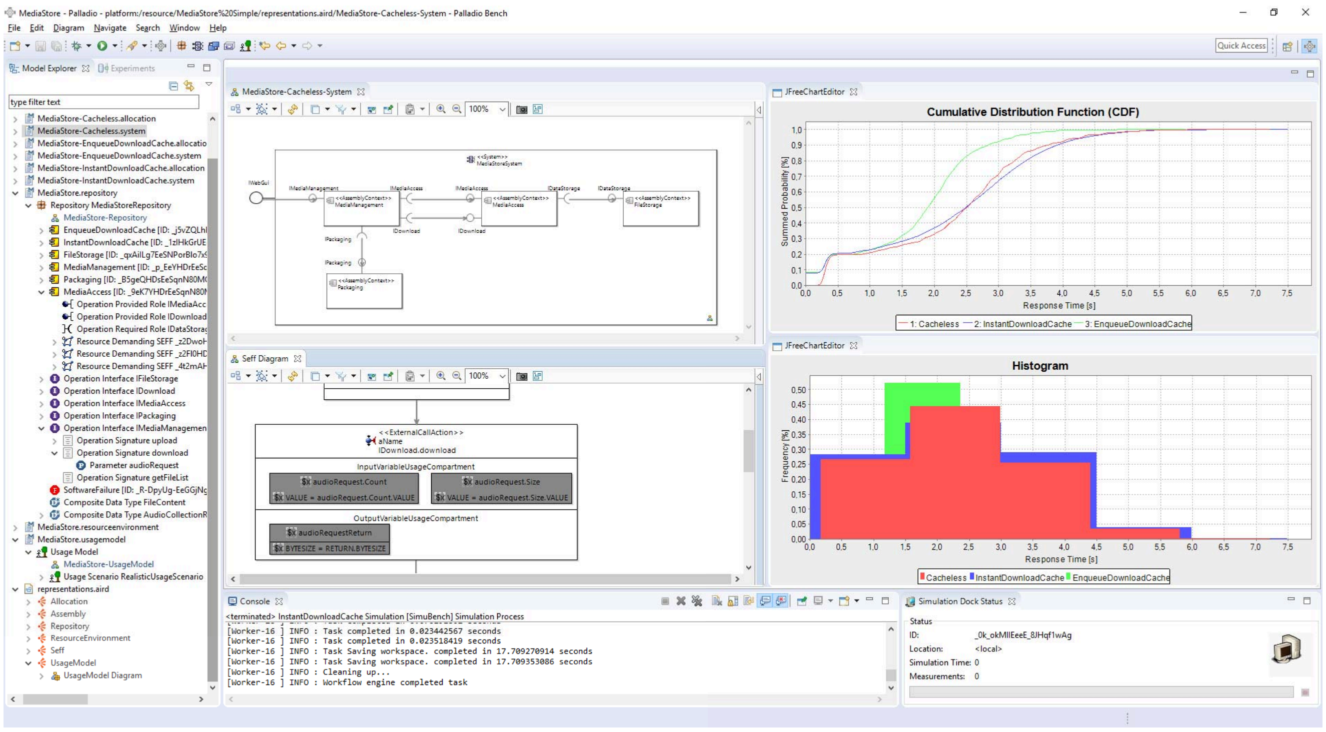
An overview of the editors and analysis views of the Palladio-Bench is given in the screenshot in Fig. 3. The figure represents the system view of a simplified Media Store in a graphical editor to specify the software architecture by wiring components from the repository model via their required and provided interfaces. The figure also depicts the Service Effect Specification view for modeling the inner behavior of components using an activity diagram like graphical editor. In the screenshot an excerpt of the Service Effect Specification for the download service is shown. Moreover, the figure shows the simulation view (Simulation Dock Status) as well as the visualization of performance simulation results for several alternatives of the Media Store's design in the experiment view. Performance simulation results of the cacheless Media Store are represented red. Results for the Media Store with instant download Cache are depicted blue and for the Media Store with enqueued download Cache green. Response times of the download service are represented for the several design alternatives as commutative distribution function and histogram.

Based on the simulation results provided by the Palladio-Bench developers can identify and compare the performance characteristics of the design alternatives. To answer aforementioned questions, the commutative distribution functions and histogram clearly show that the choice of an enqueued download Cache is not a bottleneck but decrease the average response time of the Media Store's download service.

## 5 CONCLUSION AND FUTURE DEVELOPMENT

In this paper we presented the Palladio-Bench, the world's first software architecture simulator that successfully implements the idea of an engineering discipline. With the Palladio-Bench software engineers can predict the quality properties of a software architecture based on design models instead of testing a prototype.

Palladio is continuously extended to address recent trends in software engineering discipline. Some examples of extensions are mentioned hereafter. In the CloudScale project Palladio and SimuLizar


MediaStore - Palladio - platform:/resource/MediaStore%20Simple/representations.aird/MediaStore-Cacheless-System - Palladio Bench
File Edit Diagram Navigate Search Window Help
Model Explorer
Experiments
MediaStore-Cacheless-System
Seff Diagram
JFreeChartEditor
Cumulative Distribution Function (CDF)
Summed Probability [%]
Response Time [s]
1: Cacheless 2: InstantDownloadCache 3: EnqueueDownloadCache
Histogram
Frequency [%]
Response Time [s]
Cacheless InstantDownloadCache EnqueueDownloadCache
<<ExternalCallAction>>
aName
IDownload.download
InputVariableUsageCompartment
audioRequest.Count
VALUE = audioRequest.Count.VALUE
audioRequest.Size
VALUE = audioRequest.Size.VALUE
OutputVariableUsageCompartment
audioRequestReturn
BYTESIZE = RETURN.BYTESIZE
Console
Simulation Dock Status

```
<terminated> InstantDownloadCache Simulation [SimuBench] Simulation Process
[Worker-16 ] INFO : Task completed in 0.023442567 seconds
[Worker-16 ] INFO : Task completed in 0.023518419 seconds
[Worker-16 ] INFO : Task Saving workspace. completed in 17.709270914 seconds
[Worker-16 ] INFO : Task Saving workspace. completed in 17.709353086 seconds
[Worker-16 ] INFO : Cleaning up...
[Worker-16 ] INFO : Workflow engine completed task
```

Status
ID: _0k_okMIlEeeE_8JHqf1wAg
Location: <local>
Simulation Time: 0
Measurements: 0

**Figure 3: Overview of the Editors in the Palladio Bench**

have been extended with dedicated metrics for elasticity and costs of cloud computing systems [5]. The CACTOS project enriched Palladio by energy consumption analysis [11]. The use of Palladio models for quality analysis during system operations in the DevOps context have been investigated in the iObserve project [4].

One way of future development is extending the modeling language to address current limitations like missing specification and analysis of proactive components, main memory consumption and network details. Furthermore, constructs for modeling characteristics of cloud computing and recent execution environments including global distributed elastic load balancers, auto scaling groups, NoSQL databases and eventual consistency are topics of future development. Besides modeling language extension, additional quality properties like privacy could be supported. Performance measurement and modeling must be more tightly integrated, e.g. by creating performance models automatically derived while observing the executed system. Support for sensitivity analysis could be improved where model parameters are varied to learn their influence.

## REFERENCES

[1] Matthias Becker, Markus Luckey, and Steffen Becker. 2013. Performance Analysis of Self-adaptive Systems for Requirements Validation at Design-time. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures (QoSA '13)*. ACM, New York, NY, USA, 43–52.

[2] Steffen Becker. 2008. *Coupled Model Transformations for QoS Enabled Component-Based Software Design*. Karlsruhe Series on Software Quality, Vol. 1. Universitätsverlag Karlsruhe.

[3] F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner. 2012. Architecture-Based Reliability Prediction with the Palladio Component Model. *IEEE Transactions on Software Engineering* 38, 6 (2012), 1319–1339.

[4] Robert Heinrich. 2016. Architectural Run-time Models for Performance and Privacy Analysis in Dynamic Cloud Applications. *SIGMETRICS Perform. Eval. Rev.* 43, 4 (2016), 13–22. https://doi.org/10.1145/2897356.2897359

[5] Sebastian Lehrig. 2014. Applying Architectural Templates for Design-Time Scalability and Elasticity Analyses of SaaS Applications. In *Proceedings of the 2Nd International Workshop on Hot Topics in Cloud Service Scalability (HotTopiCS '14)*. ACM, Article 2, 8 pages.

[6] Sebastian Lehrig and Steffen Becker. 2015. The CloudScale Method for Software Scalability, Elasticity, and Efficiency Engineering: A Tutorial. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE '15)*. ACM, 329–331.

[7] Philipp Merkle and Jörg Henss. 2011. EventSim – An Event-driven Palladio Software Architecture Simulator. In *Palladio Days 2011 (Karlsruhe Reports in Informatics ; 2011,32)*. KIT, 15–22. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000025188

[8] C. Rathfelder and B. Klatt. 2011. Palladio Workbench: A Quality-Prediction Tool for Component-Based Architectures. In *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*. 347–350.

[9] Ralf H. Reussner, Steffen Becker, Jens Happe, Robert Heinrich, Anne Koziolek, Heiko Koziolek, Max Kramer, and Klaus Krogmann. 2016. *Modeling and Simulating Software Architectures – The Palladio Approach*. MIT Press.

[10] Kiana Rostami, Johannes Stammel, Robert Heinrich, and Ralf Reussner. 2015. Architecture-based Assessment and Planning of Change Requests. In *11th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 21–30.

[11] Christian Stier, Henning Groenda, and Anne Koziolek. 2014. *Towards Modeling and Analysis of Power Consumption of Self-Adaptive Software Systems in Palladio*. Technical Report. University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology. 18 pages.

[12] Misha Strittmatter and Amine Kechaou. 2016. *The Media Store 3 Case Study System*. Technical Report 2016,1. Faculty of Informatics, Karlsruhe Institute of Technology. http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/3792054

[13] Emre Taspolatoglu and Robert Heinrich. 2016. Context-based Architectural Security Analysis. In *13th Working IEEE/IFIP Conference on Software Architecture*. IEEE, 281–282.