An Empirical Investigation of the Applicability of a Component-Based Performance Prediction Method

Anne Martens[1], Steffen Becker[2], Heiko Koziolek[3], and Ralf Reussner[1]

[1] Chair for Software Design and Quality
Am Fasanengarten 5, University of Karlsruhe (TH), 76131 Karlsruhe, Germany
[2] FZI Forschungszentrum Informatik
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
[3] ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Germany
{martens,sbecker,koziolek,reussner}@ipd.uka.de

Abstract. Component-based software performance engineering (CBSPE) methods shall enable software architects to assess the expected response times, throughputs, and resource utilization of their systems already during design. This avoids the violation of performance requirements. Existing approaches for CBSPE either lack tool support or rely on prototypical tools, who have only been applied by their authors. Therefore, industrial applicability of these methods is unknown. On this behalf, we have conducted a controlled experiment involving 19 computer science students, who analysed the performance of two component-based designs using our Palladio performance prediction approach, as an example for a CBSPE method. Our study is the first of its type in this area and shall help to mature CBSPE to industrial applicability. In this paper, we report on results concerning the prediction accuracy achieved by the students and list several lessons learned, which are also relevant for other methods than Palladio.

Keywords: Performance Prediction, Empirical Study, Controlled Experiment.

1 Introduction

A benefit of component-based development is the possibility to reason on properties of the complete systems based on component specifications supplied by individual component developers. With this approach, it is possible for software architects to assess the functional and extra-functional (e.g., performance, reliability) properties of a component-based system during early development stages. To do so, software architects combine component specifications to form architecture specifications. The specifications are design models (e.g. in UML) annotated with performance properties. After modelling the architecture, software architects can check performance predictions from tools analysing the architecture specifications against their requirements. This may avoid implementing designs with poor extra-functional properties and prevent subsequent costs for restructuring an implementation after detecting design-related flaws.

Researchers have developed several methods in this context, which aim at performance (i.e., response times, throughput, resource utilisation) predictions for component-based designs [4]. However, there are few real-life case studies involving

N. Thomas and C. Juiz (Eds.): EPEW 2008, LNCS 5261, pp. 17–31, 2008.

these compo-nent-based methods, which still lack industrial maturity. Several methods (e.g., [8,10,20]) simply lack tool support, while other methods (e.g., [5,6,7,23]) rely on prototypical implementations, which only have been used by their authors and require specialist knowledge. Therefore, their applicability in an industrial setting involving typical developers is unknown. Further methods for component-based performance analysis are outside the scope of this paper as they are measurement-based (i.e. predictions are based on observation of the implemented system's performance) and do not target early design stages, and also do not involve reusable component performance specifications (e.g., [13,15]).

To investigate the applicability, we conducted a controlled experiment with 19 computer science students, who analysed the performance of two different component-based designs using the Palladio method [4] as an example for a CBSPE method. We also let the students apply the well-known SPE method [21], which is not specific for component-based systems, on the same designs and compared the results. The study involved training the students in the methods and the accompanying tools as well as designing several architectural design alternatives for the analysed systems, which the students evaluated for their performance properties.

In a former paper [17], we reported on results concerning the effort needed by the students to model and analyse the system. We found that the effort was less than twice as high as for a method without reusable, component-based performance specifications (i.e., the SPE method). Therefore, the effort of creating a component performance specification could already be justified, if the component and its performance model is reused at least once. For reasons of self-containedness, sections 2, 3.2 - 3.4, 5 and 6 are similar in both papers, as they describe and discuss the common experiment setting.

For this paper, we have analysed the data collected during the experiment further (also see [16]). We focus on the accuracy of the predictions achieved by the students compared to a sample solution. Additionally, we searched for reasons for the achieved prediction accuracy by analysing the models created during the experiment and evaluating questionnaires filled out by the participants after the experiment. While the results have been obtained for a single CBSPE method, they are also interesting for the authors of other CBSPE methods. Therefore, we describe lessons learned during the study.

The contributions of this paper are (i) experimental results about the prediction accuracy achieved by third-party users of a CBSPE method and (ii) a quantitative and qualitative analysis for the reasons that led to the achieved prediction accuracy. Our study is the first of its type in this area, as we are not aware of any other studies on a CBSPE method being applied by third-party users. This may be a result of the novelty of these methods. The study helps to bring CBSPE closer to industrial maturity and is an important prerequisite for large scale industrial case studies.

The paper is organised as follows. Section 2 briefly describes the Palladio performance prediction method, so that the reader can assess the experimental tasks. Section 3 explains the goals, questions, hypotheses, and metrics used in this experiment according to the GQM paradigm [3] and describes the experimental design and conduction. Section 4 first lists the results for the formerly defined metrics collected in this experiment and afterwards discusses lessons learned. Section 5 includes potential

threats to the validity of our study to round up the experimental description. Section 6 lists related work to this study, before Section 7 concludes the paper.

2 Palladio Component Model

The Palladio Component Model (PCM) [5,19] is a meta-model for specifying and analysing component-based software architectures with focus on performance prediction.

This meta-model is divided among the separate developer roles of a component-based development process, providing each role with a domain-specific language suited to capture their specific knowledge [5]. The language of component developers targets at producing independent, reusable component specifications, that are parametrised by influence factors whose later values are unknown to the component developer. In particular, these are (i) the performance measures of external service calls, which depend on the actual binding of the component's required interfaces (provided by the software architect in the assembly model), (ii) the actual resource demands which depend on the allocation of the components to hardware resources (provided by the system deployer), and (iii) performance-relevant input/output parameters of service calls (provided by the domain expert in the usage model).

The parametric behavioural specification used in the PCM as part of the software model is the *Resource Demanding Service Effect Specification* (RD-SEFF) which is a control and data flow abstraction of single component services. It specifies control flow constructs like loops or branches if they affect external service calls. Additionally, it abstracts component internal computations in so called *internal actions* which only contain the resource demand of the action but not its concrete behaviour. Calling services and parameter passing are specified using *external call actions*, which only refer to the component's required interfaces to stay independent of the component binding.

Tool support. The PCM is supported by the PCM-Bench (see [19]), which is based on the Eclipse platform and provides UML-like graphical editors for PCM instances. For performance annotations, it uses a textual syntax, providing editors which help entering the expression with auto-completions, type-checking and syntax highlighting. OCL is applied to increase completeness and correctness of PCM model instances. A simulation tool predicts performance measures of the G/G/n queueing system a PCM instance represents. It uses specialised queueing networks as the performance model and is generated from a PCM instance using model transformations.

The resulting prediction metrics are response time distributions of single external service calls as well as for a whole scenario. They are visualised as cumulative distribution functions (CDFs) or histograms. The utilisation of resources is visualised using pie charts.

3 Empirical Investigation

For the empirical investigation, we formulated a goal, two question and derived metrics using the Goal-Question-Metric approach [3] The goal of this work is to *empirically*

evaluate the applicability of the Palladio approach from a third-party user's point of view.

The same metrics can also be used when repeating this experiment, also for other approaches. In this paper, we focus on the results for the achieved accuracy when Palladio is applied by third-party users. Details of the concerned two questions, their hypotheses, and their metrics are presented in section 3.1. For comparison, with the same question and metrics, we also investigated the SPE approach [21], which offers no special support for component-based systems. For brevity, we keep the presentation of the SPE results short and focus on the results for Palladio.

We conducted the investigation as a controlled experiment. Section 3.2 presents the experiment's design, section 3.3 describes the preparation of the participants, and section 3.4 presents the systems under study.

3.1 Questions and Metrics

Due to space limitations, only informal explanations of the metrics are given here. The formal definitions can be found in [16, p.35]. Table 1 summarises questions and metrics.

To study the applicability of Palladio, we first carefully created performance models of the systems under study ourselves as sample solutions. These sample solutions are unique for the information provided in the experimental task, as adding any information or omitting any information from the experimental task would not reflect the system properly any more. During the experiment, we gave the students enough information to create performance models for the different design alternatives themselves. Afterwards, we assessed the participants' models by comparing their prediction to predictions from the sample solution. Thus, in the following, *quality of the models* is defined to be the similarity to the sample solution. We measured the applicability in terms of how well the participants understand the approaches and how usable the given tools are, and therefore we asked the following questions and defined the following metrics.

Q1: What is the quality of the created performance prediction models? First, a performance model should enable predictions that are similar to the reference performance model (i.e. the sample solution) when analysed. Here, the predicted response time was an important performance metric. Thus, we defined metric 1.1: *Relative deviation of predicted mean response times of the participants and of the reference model* (percentage).

To assess different design alternatives when designing or changing a system, the relation of the respective response times is also of interest. We let the participants evaluate

Table 1. Summary GQM Questions and Metrics

Question 1	**What is the quality of the created performance prediction models?**
Metric 1.1	Relative deviation of predicted mean response times of the participants and of the reference model.
Metric 1.2	Percentage of correct design decisions.
Metric 1.3	Normalised deviation in design decision rankings.
Question 2	**What are the reasons for potentially deviating predictions?**
Metric 2.1	Problems when creating the models and classification

several design alternatives and measured how many participants correctly identified the best design alternative in respect of its response time by stating metric 1.2: *Percentage of correct design decisions*.

As a software architect does not necessarily choose the design alternative with the best performance, but might consider other quality attributes or cost, the results for the performance-wise inferior design alternatives are also important. Thus, next to identifying the best design alternative, the participants had to rank all alternatives. The ranking of design alternatives by the participants was compared to the ranking of the design alternatives of the reference solution in metric 1.3: *Normalised deviation in design decision rankings*. For this metric, we counted how many ranks lie between the position of a design alternative in the ranking of a participant and the correct position of a this design alternative in the ranking for the reference solution. We normalised this metric so that a correct ranking has a deviation of 0% and the reversed ranking a deviation of 100%. Additionally, we recognised very similar response times as virtually equal design alternatives and did not punish rankings that permuted them.

Our hypothesis 1 was that (1) the average deviation as measured with metric 1.1 is not larger than 10%, (2) 80% of the participants can choose the correct design decision and (3) the rankings deviate no more than 10% in average for both Palladio and SPE.

Q2: What are the reasons for potentially deviating predictions? Several factors might influence the quality of a prediction. First of all, the participants need to understand the approaches and their various concepts. Additionally, the tools has to be usable and support an easy creation and maintenance of the models. Problems in both areas could lead to modelling errors and therefore to erroneous predictions. Next to modelling problems, errors in interpreting the prediction results might lead to false conclusions. This depended on the results the approach gave as well as on visualisation of results in the tool.

To measure the problems, we documented questions of the participants and errors in the final models, that appeared during the acceptance test or were found in the final models. Each such question or error is counted as one problem in metric M2.1: *Problems when creating the models and classification*.

Our hypothesis 2 was that most problems arise from a lack of understanding and tool difficulties.

3.2 Experiment Design

The study was conducted as a controlled experiment. The participants of this study were students of a master's level course. In an experiment, it is desirable to trace back the observations to changes of one or more independent variables. Therefore, all other variables influencing the results need to be controlled. The *independent variable* in this study was the prediction approach (i.e. Palladio or SPE). Observed *dependent variables* were the quality of the created models in terms of similarity with a reference model and the problems occurring during the experiments or being detected in the final models.

The experiment was designed as a cross-over trial [12] as depicted in figure 1. The participants were divided into two groups, each applying an approach to a given task. In a second session, the groups applied the other approach to a new task. Thus, each

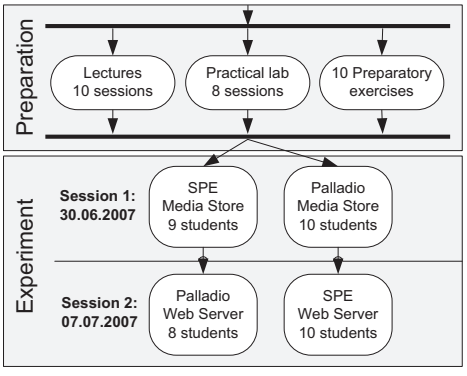

Fig. 1. Experiment design

participant worked on two tasks in the course of the experiment (inter-subject design) and used both approaches. This allowed us to collect more data points and balanced potential differences in individual factors like skill and motivation between the two experiment groups. Additionally, using two tasks lowered the concrete task's influence and increased the generalisability.

We balanced the grouping of the participants based on the results in the preparatory exercises: We divided the better half randomly into the two groups, as well as the less successful half, to ensure that the groups were equally well skilled for the tasks. We chose not to use a counter-balanced experiment design, as we would have needed to further divide the groups. In that case, the groups would have been to small and the individual's performance would have been too much an influence. We expected a higher threat to validity from the individual participant's performance than from sequencing effects (also called carry-over effects, [12]).

Before handing in, the participants' solutions were checked for minimum quality (less than 50% deviation) by comparing the created models to the respective reference model. This acceptance test included the comparison of the predicted response time with the reference model's predicted response time as well as a check for the models' well-formedness.

3.3 Student Teaching

The 19 computer science students participating in the experiment were trained in applying Palladio and SPE in a course covering both theory and practical labs. For the theory part, there was a total of ten lectures, each of them took 1.5h. The first three lectures were dedicated to foundations of performance prediction and CBSE. Then, two lectures introduced SPE followed by five lectures on Palladio. The three additional lectures on Palladio in comparison to SPE were due to its more complex meta-model which allows for reusable prediction models. In parallel to the lectures, eight practical labs took place, again, each taking 1.5h. During these sessions, solutions to the accompanying ten exercises were presented and discussed. Five of these exercises practised SPE and five Palladio.

The exercises had to be solved by the participants between the practical labs. We assigned pairs of students to each exercise and shuffled the pairs frequently to get different combinations of students work together and exchange knowledge. Each exercise took the students 4.75h in average to complete.

Overall, the preparation phase was intended to ensure a certain level of familiarity with the tools and concepts, because participants who failed two preparatory exercises or an intermediate short test were excluded from the experiment.

3.4 Experiment Tasks

To be applicable for both SPE and Palladio, the experiment tasks can only contain aspects that can be realised with both approaches. For example, the tasks did not make use of the separate developer roles of Palladio.

Both experiment tasks had similar set-ups. The task descriptions contained component and sequence diagrams documenting the static and dynamic architecture of a component-based system. The sequence diagrams also contained performance annotations. The resource environment with servers and their performance properties was documented textually. The detailed task description is available on-line in [16]. For each system, two usage scenarios were given, to reflect both a single-user scenario (*UP1*) and a multi-user scenario leading to contention effects (*UP2*). Additionally, they differed in other performance relevant parameters (see below).

In addition to the initial system, several design alternatives were evaluated. Four of them were designed to improve the system's performance, and the participants were asked to evaluate which alternative is the most useful one. Three of these alternatives implied the creation of a new component, one only changed the allocation of the components and the resource environment by introducing a second machine. With the final fifth alternative, the impact of a change of the component container, namely the introduction of a broker for component lookups, on the performance should be evaluated.

The systems in both tasks were prototypical component-based systems. In the first session, a performance prediction for a web-based system called Media Store was conducted. This system stores music files in a database. Users can either upload or download sets of files. The size of the music files and the number of files to be downloaded are performance-relevant parameters. The five design alternatives were the introduction of a cache component that kept popular music files in memory (v_1^{MS}), the usage of a thread pool for database connections (v_2^{MS}), the allocation of two of the components to a second machine (v_3^{MS}), the addition of a component that reduces the bit rate of uploaded files to reduce the file sizes (v_4^{MS}) and the aforementioned usage of a broker (v_5^{MS}).

In the second session, a prototypical Web Server system was examined. Here, only one use case was given, a request of an HTML page with further requests of potential embedded multimedia content. Performance-relevant parameters were the number of multimedia objects per page, the size of the content and the proportion of static and dynamic content. The five design alternatives were the introduction of a cache component (v_1^{WS}), the aforementioned usage of a broker (v_2^{WS}), the parallelisation of the Web Server's logging (v_3^{WS}), the allocation of two of the components on a second machine (v_4^{WS}) and the usage of a thread pool within the Web Server (v_5^{WS}).

The participants who used the Palladio approach were provided with an initial repository of available components and their interfaces, but not their behavioural description (i.e., RD-SEFFs, see section 2). It made the tasks for SPE and Palladio more comparable, because the participants still had to create the RD-SEFFs with the performance annotations, which is similar to the creation of an SPE model.

4 Results

In this section, we interpret the measured data based on the GQM plan. The structure of this section follows the two questions, each being partitioned into the presentation of the metrics. In the paper, we only present the evaluation of the metrics for Palladio. The results for SPE can be found in [16, p.83]. The metrics are evaluated for both tasks. Finally, the hypothesis of each question is checked based on the measured metrics.

4.1 What is the Quality of the Created Performance Prediction Models?

Metric 1.1: Relative deviation of predicted mean response times between the participants and the reference model. Table 2 shows the results of metric 1.1 for Palladio.

We first consider the average deviation for each task. Overall, the deviation is lower using the Media Store and for $UP1$. The overall average is low with 6.9%. Interestingly, the deviation varied a lot between the different design alternatives. For the Media Store and Palladio, the alternative v_3^{MS} (second server), has a high deviation, and v_0^{MS} for the $UP2$, too. For the Web Server and Palladio, the deviations for the v_2^{WS}, the broker alternative, v_0^{WS}, v_1^{WS} (Cache), and v_3^{WS} (Logging) are also high.

For SPE, we measured a slightly higher average deviation of 8.3% and also strong variations for the different design alternatives.

Metric 1.2: Percentage of correct design decisions. For metric 1.2, we compared the results of the reference model (cf. section 3.1) with the participants rankings and assessed the percentage of correct identification of the performance-wise best design alternative. Some participants did not manage to model all alternatives in the given time and thus, their rankings were incomplete and their results cannot be used (see fig. 1 for the total numbers of participants).

As the predicted response time of the best and second-best alternatives of the Media Store were close to each other, we made no distinction between these two. Thus, all participants chose right, because all of them identified either the bit rate (v_4^{MS}) or the

Table 2. Metric 1.1: Relative deviation of the predicted response times for Palladio

		v_0^s	v_1^s	v_2^s	v_3^s	v_4^s	v_5^s	Avg
Media Store	$UP1$	1.93%	0.90%	0.49%	20.08%	3.02%	1.69%	4.69%
$(s = MS)$	$UP2$	13.21%	2.20%	4.15%	13.23%	4.42%	3.51%	6.79%
Web Server	$UP1$	1.00%	11.07%	1.94%	4.23%	4.55%	9.40%	5.47%
$(s = WS)$	$UP2$	15.92%	20.35%	10.87%	10.67%	2.57%	3.64%	10.67%
Overall $propDevMeanResp_{Pal}$								6.90%

cache option (v_1^{MS}) as the best design alternative and ranked the respective other one second-best.

For the Web Server, $UP1$ and Palladio, 4 out of 6 participants who ranked all alternatives identified the second server v_4^{WS} as the best alternative. Of the two others, one actually predicted a lower response time for the cache (v_1^{WS}), the other one seemed to have other reasons or could not correctly interpret the CDF, as the second server v_4^{WS} is faster for his model, too. We get $perc_{WS,UP1,Pal} = 0.67$. All eight SPE participants chose the right alternative: $perc_{WS,UP1,SPE} = 1$.

For usage model 2, all five Palladio participants who ranked all alternatives identified the second server v_4^{WS} as the best alternative. For SPE, 7 out of 8 participants who ranked all alternatives did so: $perc_{WS,UP2,SPE} = 0.88$.

Combined[1] we get $perc_{SPE}$ = 0.97 and $perc_{Pal}$ = 0.85.

Metric 1.3: Normalised deviation in design decision rankings. Not all participants ranked all alternatives, because they did not complete all predictions or missed the time to complete the ranking, even if they completed the predictions. We still used the incomplete rankings for the evaluation of the metrics, but were careful to weight complete rankings stronger (cf. [16, p.86f]).

For Palladio, the ranks were wrong by 6.5% of the maximum possible permutation. For SPE, the ranks were wrong by 7.3% of the maximum possible permutation. Thus, SPE rankings were more permuted by factor 0.12 compared to Palladio rankings.

Hypothesis 1. With both approaches, the mean response time predicted by the participants only deviates in average 6.9% (Palladio) and 8.3% (SPE) from the mean response time predicted for the reference model. Thus, the deviation of the average is within the limit of 10%. However, for single alternatives, the deviation was higher (see table 2). These pose a threat to hypothesis 1.

Most participants also were able to identify the correct design decisions, in particular 85% for Palladio and 97% for SPE, both is within the bounds of 80%. Finally, the deviation of the ranking is also low (not more than 10% in average).

Overall, the results indicate that hypothesis 1 cannot be rejected for the average case. However, the high variation of the deviation of the predicted mean response time between the different design alternatives hampers assessing hypothesis 1. As the alternatives have differing results, it is unclear how the metrics would be evaluated for different design alternatives.

4.2 What Are the Reasons for Potentially Deviating Predictions?

Metric 2.1: Number of problems and classification. Table 3 shows the problems in the different areas for Palladio, first the tools, then the method itself. For the PCM Bench (i.e. the tool), we identified the problem areas of tool usage, of interpreting the error messages and of bugs of the tool. With Palladio, most problems were with the usage of the tool, e.g. participants asked how to create component parameters or a usage model. Interestingly, there were more usage problems with the Web Server task than

[1] Note that the percentages for the two systems do not equally influence the results, but are weighted by the number of decisions by definition of the metric (cf. [16, p.41])

Table 3. Metric 2.1: Average number of problems per participant for Palladio

	Tool				Methodology						
	Usage	Error	Bug	Sum	Parameters	Component parameters	Types and units	Assembly	Usage model	Sum	Sum
Media Store	0.57	0.43	0.29	1.29	1.57	1.00	0.71	0.00	0.00	3.29	4.57
Web Server	2.25	0.38	0.63	3.25	0.63	0.38	0.63	0.13	0.13	1.88	5.13
Both systems	1.41	0.40	0.46	2.27	1.10	0.69	0.67	0.06	0.06	2.58	4.85

with the Media Store task. Relatively more tool problems occurred with Palladio (in average 2.27 per participant, that is 47% of the problems) than with SPE (in average 0.24 per participant, that is 5% of the problems). Although the number of participants was relatively small, and outliers might strongly have influenced this result, we still give the average values here. No clear outliers were detected, every participant was included in both groups (because of the cross-over plan) and the effect was fairly large, thus, the average values were still meaningful.

For the Palladio method (i.e., separated from the tool), different problem areas were identified: (1) The specification of parameter values (e.g. specifying the number of requested audio files), especially (2) the specification of component parameters (e.g. specifying the size of audio files stored in the database), (3) the handling of data types (e.g. using string and enum values within the model) and annotation units (e.g. confusing seconds with millisecond within one model), (4) the assembly (wiring the components) and (5) the usage model (specifying the user flow). Here, in average most problems concerned the specification of parameter values, followed by the specification of component parameters and of types and units. Interestingly, this relation is very pronounced for the Media Store, but less pronounced for the Web Server, where there were equally many problems with parameters and types and units, followed by component parameters. The participants using Palladio for the Web Server task had more problems with the tool than with the methodology, the opposite applies to the participants using Palladio for the Media Store task. Overall, participants using Palladio had 2.58 methodology problems per participant in average (that is 53% of the problems). In comparison, participants using SPE had 4.21 methodology problems per participant in average (that is 95% of the problems). Thus, compared to SPE, participants using Palladio had more problems with the actual tool implementation and less with the methodology itself.

For Palladio, 77% of the problems occurred during the experiment and were captured in the question protocol, 12% in the acceptance test, and 11% were still present in the final models. For SPE, 30% were captured in the question protocol, 26% in the acceptance test, and 44% of the problems were still present in the final model.

Hypothesis 2. Our hypothesis 2 was that most problems arise from a lack of understanding and tool difficulties. The number of problems detected, being in average more than 4 per participant for both approaches, show that there was a significant number of problems. Still, as the quality of the created models was overall satisfactory, they do not invalidate the principle applicability of the approaches.

As expected, problems arose from a lack of understanding of the methodology and tool difficulties. Additionally, problems with the task description were detected (not included in the table 3 above).

4.3 Lessons Learned

Applicability: The participant in this study were able to create models with a good quality (as defined in section 3.1): Their predictions had a low deviation compared to a reference model, they were mostly able to choose the best design alternative and they successfully ranked the design alternatives based on the predicted performance. The results are comparable to the quality of models created with SPE. As SPE is a mature approach also applied in industry, the results suggest that also Palladio can be applied by third-party users.

Tool influence: A large fraction of the problems detected for Palladio concerned the tool, i.e. the current implementation of the approach. This supports our conviction that the tool is an important part of any study of applicability of approaches, and must be taken into account when designing and executing them.

Methodology: The results also show that there were fewer methodology problems for the component-based approach Palladio than for the mature SPE, even though the meta model is much more complex: While the SPE meta model consists of 28 classes, the Palladio meta model has about 100 classes, of which most are needed in every model. The results show that the complexity of the meta model can be hidden in the tool and does not hinder the applicability. In the qualitative questionnaire, most participants even stated to have understood the Palladio concepts better than SPE constructs [16].

Occurrence of problems: Problems occurred early during the experiment for Palladio, whereas for SPE, more problems remained in the final models as errors. This suggests that a tool support with many constraint checks against the meta model helps the user to identify problems. Thus, checks contribute to fewer errors in the final models.

Interpretation of results: We saw that distribution functions as resulting metrics are comprehensible for the users, although they were harder to interpret than the mean value and resulted in more errors. More teaching effort is required to make users familiar with the analysis results, and more effort should be spent to improve the presentation of the prediction results.

Influence of the system under study: Finally, the study detected an influence of the system under study on the applicability. Both the quality of the created models, the number of problems and the needed effort (cf. [17]) depend on the actual design decision under study. In this study, the Web Server system seems to be in general more fitted for Palladio, whereas the Media Store system seems to be more fitted

for SPE. This is also supported by qualitative results, because only some participants from the (Palladio, Media Store) group and the (SPE, Web Server) group stated that the task at hand was too difficult. All participants from the other two groups stated the task difficulty was adequate.

5 Threats to Validity

To enable the reader to assess our study, we list some potential threats to its validity in the following. We look at the internal, construct, and external validity (a more thorough discussion can be found in [16]).

The *internal validity* states whether changes of an experiment independent variables are in fact the cause for changes of the dependent variables [22, p.68]. Controlling potential interfering variables ensures a high internal validity. In our experiment, we evaluated the pre-experiment exercises and assigned the students to equally capable groups based on the results to control the different capabilities of the participants. A learning effect might be an interfering variable in our experiment, as the students finished the second experiment session faster than the first one.

A potential bias towards or against Palladio was threatening the internal validity in our experiment, as the participants knew that the experimenters were involved in creating this method. However, we did not notice a strong bias from the collected data and the filled-out questionnaires, as the participants complained equally often about the tools of both approaches.

The *construct validity* states whether the persons and settings used in an experiment represent the analysed constructs well [22, p.71]. Palladio and SPE are both typical performance prediction methods involving UML-like design models. The SPE approach has no special support for component-based systems, and was chosen for the experiment due to its higher maturity compared to existing CBSPE approaches. Additionally, SPE only supports M/M/n queueing systems and reports only mean values. We designed the experimental tasks so that not all specific features of Palladio (e.g. separation of developer roles in component-based development, performance requirements using quantiles) were used to ensure a balanced comparison.

While our experiment involved student without long-time industrial experience, we argue that their performance after the training sessions was comparable to the potential performance of practitioners. Most of the students were close to graduating and will become practitioners soon. Due to the training, their knowledge about the methods was more homogeneous than the knowledge of practitioners with different backgrounds. Studies, such as [11], suggest the suitability of students for similar experiments.

The *external validity* states whether the results of an experiment are transferable to other settings than the specific experimental setting [22, p.72]. While we used medium-sized, self-designed systems for the tasks, we modelled these system designs and the design alternatives after typical distributed systems and commonly known performance patterns [21], which are representative for the systems usually analysed in this area.

We tried to increase the external validity of our study by letting the participants analyse two different systems, so that differences in the results could be traced back to

the systems, and not the prediction methods. Effects that are observed for both tasks are thus more likely to be generalisable to other settings.

Still, the systems under study were modelled on a high abstraction level due to the time constraints of such an experiment. More complex systems would increase the external validity, but would also involve more interfering variables, thus decreasing the internal validity. Furthermore, the available information at early development stages is usually limited, which would be reflected by our experimental setting.

6 Related Work

Basics about the area of *performance prediction* can be found in [18,21]. Balsamo et al. [1] give an overview of about 20 recent approaches based on queueing networks, stochastic Petri nets, and stochastic process algebra. Becker et al. [4] survey performance prediction methods specifically targeting component-based systems. Examples are CB-SPE [6], ROBOCOP [7], and CBML [23].

Empirical studies and controlled experiments [22] are still under-represented in the field of model-based performance predictions, as hardly any studies comparable to ours can be found. Balsamo et al. [2] compared two complementary prediction methods (one based on SPA, one on simulation) by analysing the performance of a naval communication system. However, in that study, the authors of the methods carried out the predictions themselves. Gorton et al. [9] compared predicted performance metrics to measurements in a study, but only used one method for the predictions.

Koziolek et al. [14] conducted a study similar to the one presented in this paper. They compared three different performance prediction methods, which were not specific for component-based systems. The study also involved the SPE methods and attested it the most maturity and suitability for early performance predictions and influenced our decision to compare Palladio to SPE.

7 Conclusions

We have conducted a controlled experiment with 19 computer science students investigating the applicability of a CBSPE method (our Palladio method) by third parties. After several training sessions, the students modelled and analysed the performance of two different component-based designs and assessed five different design alternatives for each system. We found that the quality of the models and predictions created by the students deviated less than 10 % from the predictions achieved with a reference model created by the experimentators. Furthermore, we learned that more than 80% of students were able to rank the given design alternatives correctly. Reasons for the still existing deviations in the predictions were traced back to problems with the involved tools (47%) and to problems with the methodology (53%).

To the best of our knowledge, our experiment is the first empirical study involving a CBSPE method applied by persons other than their authors. Researchers and practitioners can benefit from this type of study. Researchers can use the lessons learned during our experiment to improve their own CBSPE methods, as these lessons are not specific for the Palladio method. For practitioners, the training material and improved

tool support created for this experiment may lower the barrier to learn a CBSPE method and conduct early performance predictions to create better software architectures.

However, our study is still a first step to rigorously assess the applicability of CBSPE methods. Similar experiments should be conducted once the tools and methodologies mature further. Future experiments should also compare different CBSPE methods against each other to evaluate their specific benefits and deficits. It would be interesting to compare the predictions to measurements of different implementations of the designs, to analyse larger designs, and to also involve practitioners in the study.

Details on the experimental settings and the results can be found in [16], available online at `http://sdq.ipd.uka.de/diploma_theses_study_theses/completed_theses`

Acknowledgements. We would like to thank Walter Tichy, Lutz Prechelt, and Wilhelm Hasselbring for their kind review of the experimental design and fruitful comments. Furthermore, we thank all members of the SDQ Chair for helping prepare and conduct the experiment. Last, but not least, we thank all students who volunteered to participate in our experiment.

References

1. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-Based Performance Prediction in Software Development: A Survey. IEEE Trans. on Softw. Eng. 30(5), 295–310 (2004)
2. Balsamo, S., Marzolla, M., Di Marco, A., Inverardi, P.: Experimenting different software architectures performance techniques: A case study. In: Proc. of WOSP, pp. 115–119. ACM Press, New York (2004)
3. Basili, V.R., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach. In: Marciniak, J.J. (ed.) Encyclopedia of Software Engineering - 2 Volume Set, pp. 528–532. John Wiley & Sons, Chichester (1994)
4. Becker, S., Grunske, L., Mirandola, R., Overhage, S.: Performance Prediction of Component-Based Systems: A Survey from an Engineering Perspective. In: Reussner, R., Stafford, J.A., Szyperski, C.A. (eds.) Architecting Systems with Trustworthy Components. LNCS, vol. 3938, pp. 169–192. Springer, Heidelberg (2006)
5. Becker, S., Koziolek, H., Reussner, R.: Model-based Performance Prediction with the Palladio Component Model. In: Proc. of WOSP, February5–8, 2007, pp. 54–65. ACM Sigsoft, New York (2007)
6. Bertolino, A., Mirandola, R.: CB-SPE Tool: Putting Component-Based Performance Engineering into Practice. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) CBSE 2004. LNCS, vol. 3054, pp. 233–248. Springer, Heidelberg (2004)
7. Bondarev, E., Chaudron, M.R.V., de Kock, E.A.: Exploring performance trade-offs of a JPEG decoder using the DeepCompass framework. In: Proc. of WOSP 2007, pp. 153–163. ACM Press, New York (2007)
8. Eskenazi, E., Fioukov, A., Hammer, D.: Performance Prediction for Component Compositions. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) CBSE 2004. LNCS, vol. 3054, pp. 280–293. Springer, Heidelberg (2004)
9. Gorton, I., Liu, A.: Performance Evaluation of Alternative Component Architectures for Enterprise JavaBean Applications. IEEE Internet Computing 7(3), 18–23 (2003)

10. Hamlet, D., Mason, D., Woit, D.: Component-Based Software Development: Case Studies, March 2004. Series on Component-Based Software Development, chapter Properties of Software Systems Synthesized from Components, vol. 1, pp. 129–159. World Scientific, Singapore (2004)
11. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects - A comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering 5(3), 201–214 (2000)
12. Jones, B., Kenward, M.G.: Design and Analysis of Cross-over Trials, 2nd edn. CRC Press, Boca Raton (2003)
13. Kounev, S.: Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets. IEEE Trans. of SE 32(7), 486–502 (2006)
14. Koziolek, H., Firus, V.: Empirical Evaluation of Model-based Performance Predictions Methods in Software Development. In: Reussner, R., Mayer, J., Stafford, J.A., Overhage, S., Becker, S., Schroeder, P.J. (eds.) QoSA 2005. LNCS, vol. 3712, pp. 188–202. Springer, Heidelberg (2005)
15. Liu, Y., Fekete, A., Gorton, I.: Design-Level Performance Prediction of Component-Based Applications. IEEE Transactions on Software Engineering 31(11), 928–941 (2005)
16. Martens, A.: Empirical Validation of the Model-driven Performance Prediction Approach Palladio. Master's thesis, Universität Oldenburg (November 2007), http://sdq.ipd.uka.de/diploma_theses_study_theses/completed_theses
17. Martens, A., Becker, S., Koziolek, H., Reussner, R.: An empirical investigation of the effort of creating reusable models for performance prediction. In: CBSE 2008, Karlsruhe, Germany (accepted, 2008)
18. Menasce, D., Almeida, V., Dowdy, L.: Performance by Design. Prentice Hall, Englewood Cliffs (2004)
19. The Palladio Component Model, http://palladio-approach.net
20. Sitaraman, M., Kuczycki, G., Krone, J., Ogden, W.F., Reddy, A.L.N.: Performance Specification of Software Components. In: Proceedings of the 2001 symposium on Software reusability: putting software reuse in context, pp. 3–10. ACM Press, New York (2001)
21. Smith, C.U., Williams, L.G.: Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley, Reading (2002)
22. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers, Norwell (2000)
23. Wu, X., Woodside, M.: Performance Modeling from Software Components. SIGSOFT SE Notes 29(1), 290–301 (2004)