Diploma Thesis

Empirical Validation of the Model-driven Performance Prediction Approach Palladio

Anne Martens, Mat.-Nr. 8146070

14th November 2007

First examiner    Prof. Dr. Wilhelm Hasselbring
Second examiner   Prof. Dr. Ralf H. Reussner
To estimate the consequences of design decisions is a crucial element of an engineering discipline. Model-based performance prediction approaches target the estimation of a system’s performance at design time. Next to accuracy, the approaches also need to be validated for their applicability to be usable in practice.

The applicability of the model-based performance prediction approach Palladio was never validated before. Previous case studies validating Palladio were concerned with the accuracy of the predictions in comparison to measurements.

In this thesis, I empirically validated the applicability of Palladio and, for comparison, of the well-known performance prediction approach SPE. While Palladio has the notion of a component, which leads to reusable prediction models, SPE makes not use of any componentisation of the system to be analysed. For the empirical validation, I conducted an empirical study with 19 computer science students. The study was designed as a controlled experiment to achieve a high validity of the results.

The results showed that both approaches were applicable, although both have specific problems. Furthermore, it was found that the duration of conducting a prediction using Palladio was significantly higher than duration using SPE, however, the influence of potential reuse of the Palladio models was excluded by the experiment design.
# Contents

1 Introduction ........................... 1
   1.1 Motivation ................................ 1
   1.2 Contribution ................................ 4
   1.3 Related Work ................................ 5
   1.4 Structure of this Thesis ...................... 5

2 Performance Prediction .................. 7
   2.1 Theoretical Foundations ....................... 7
   2.2 Performance Prediction for Component-based Systems ...................... 11
      2.2.1 The Palladio Approach ...................... 12
      2.2.2 Other Component-based Prediction Approaches ...................... 16
   2.3 Monolithic Prediction Approaches ............... 20
      2.3.1 The SPE Approach ...................... 21
      2.3.2 Comparability of the Approaches ...................... 24

3 Research Method ....................... 29
   3.1 Empirical Studies in Software Engineering ...................... 29
      3.1.1 Controlled Experiment ...................... 31
      3.1.2 Related Empirical Studies ...................... 32
   3.2 Goal-Question-Metric Plan ...................... 33
      3.2.1 Goal of the Experiment ...................... 34
      3.2.2 Questions and Derived Metrics ...................... 35

4 Design and Conduction of the Experiment ........... 53
   4.1 Participants ............................. 53
      4.1.1 Preparation ............................. 57
      4.1.2 Preparatory Exercises ...................... 58
      4.1.3 Results of the Preparation ...................... 58
   4.2 The Experiment ............................ 59
      4.2.1 Experiment Plan ............................ 59
      4.2.2 Experimental Tasks ............................ 61
      4.2.3 Execution of the Experiment ...................... 72
   4.3 Validity of this Experiment ................. 75
      4.3.1 Conclusion Validity ...................... 77
      4.3.2 Internal Validity ............................ 77
      4.3.3 Construct Validity ...................... 79
      4.3.4 External Validity ............................ 79
5 Results

5.1 Results of the Metrics

5.1.1 What is the quality of the created performance prediction models? 83
5.1.2 What are the reasons for the model’s quality? 89
5.1.3 What is the duration of predicting the performance? 101
5.1.4 What are the reasons for the duration? 104

5.2 Discussion of the Results

5.2.1 Differences of the Approaches 109
5.2.2 Differences of the Systems under Study 115
5.2.3 Further Assessment of the Validity 117

6 Conclusions and Outlook

6.1 Summary 121
6.2 Knowledge gained 122
6.3 Future Work 123

List of Figures 124
List of Tables 126
Bibliography 128

A Tutorial Slides and Preparatory Exercises

A.1 Introductory Tutorial Slides I
A.2 SPE-ED Tutorial Slides XVIII
A.3 Palladio Tutorial Slides XXXV
A.4 Review Slides LXVIII
A.5 Preparatory Exercises CXXXVI
A.6 Durations of Solving Preparatory Exercises

B Experimental Material

B.1 Experiment Tasks CXXXVIII
B.1.1 Media Store CXXXVIII
B.1.2 Web Server CLXXII
B.2 Rank Estimation CCX
B.2.1 Media Store CCX
B.2.2 Web Server CCXII
B.3 Time Stamps CCXIV
B.3.1 Media Store CCXIV
B.3.2 Web Server CXXX
B.4 Qualitative Questionnaires CXXXVI
B.4.1 Questionnaire Media Store CXXXVI
B.4.2 Questionnaire Web Server CXXXIX
B.4.3 Comparing Questionnaire CXXXII
B.5 Acceptance Tests CCXL
B.5.1 Check Lists CCXL
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.5.2</td>
<td>Acceptance Test Protocols</td>
</tr>
<tr>
<td>B.6</td>
<td>Question Protocol</td>
</tr>
<tr>
<td>B.6.1</td>
<td>Question Protocol Sheets</td>
</tr>
<tr>
<td>C</td>
<td>Resulting Data</td>
</tr>
<tr>
<td>C.1</td>
<td>Predictions of the Participants</td>
</tr>
<tr>
<td>C.2</td>
<td>Duration and Break Down</td>
</tr>
<tr>
<td>C.3</td>
<td>Problems</td>
</tr>
<tr>
<td>C.3.1</td>
<td>Record of Problems</td>
</tr>
<tr>
<td>C.3.2</td>
<td>Cumulated Data</td>
</tr>
<tr>
<td>C.4</td>
<td>Answers to Questionnaire</td>
</tr>
<tr>
<td>C.4.1</td>
<td>Question 17</td>
</tr>
<tr>
<td>C.4.2</td>
<td>Question 21</td>
</tr>
<tr>
<td>C.4.3</td>
<td>Question 29</td>
</tr>
<tr>
<td>C.4.4</td>
<td>Question 31</td>
</tr>
</tbody>
</table>

### Acknowledgements

CCLXXVIII
1 Introduction

In this introduction, I first motivate why an empirical validation of proposed performance prediction approaches is necessary for an engineering discipline such as software engineering. After that, I account for performance prediction and its application in modern software engineering and in particular for component-based systems. Thirdly, I describe the contribution of this thesis. Finally, I present where related work can be found and give an overview on the structure of this thesis.

1.1 Motivation

Software engineering is commonly understood as being the ”systematic creation, evaluation and maintenance of systems” [IEE90]. For a systematic approach, it is crucial to make the properties of software development processes and artefacts predictable. Otherwise, if properties are not predictable, design is meaningless, as any orientation of the design towards a goal or requirements would be senseless.

From theorists, many formal methods and tools have been suggested to improve software engineering and prediction of properties, but they are largely are not applied [ER03, p.1]. There results a gap between theory and practice. This gap can be closed if practice is used more as a measure of a method’s usefulness [ER03, p.1].

To assess the usefulness of methods for software engineering, they mostly need to be validated empirically [Pre01, p.30]. In that, not only their theoretical correctness needs to be studied, but also their usability in practical applications. Only if being usable, new methods can actually be used. The term “usable” here comes with two meanings: First, a method and accompanying tools must be usable in the sense of applicable, i.e. users must be able to use them. Second, the use of the methods should also bring advantages, e.g. fasten the development process.

Overall, three types of empirical validation can be differentiated [FER08], of which type I and II have been discussed above:

**Type I validations** ”demonstrate that predictions made by a prediction method conform to the observed reality given that the method and its tools are applied without making any mistakes.” [Bec08]

**Type II validations** ”show that methods, which depend on human interaction, can be applied by trained users successfully.” [Bec08] Thus, a type II validation shows the above-mentioned first meaning of “usable” in terms of “applicable”.
Type III validations "finally seek to validate that new methods are superior to existing ones. The last type is extremely hard to show and cost-intensive in larger contexts as it requires to perform projects at least twice - one time using the method under validation and the other time without it." [Bec08] Thus, a type III validation shows the above-mentioned second meaning of "usable" in terms of "advantageous".

For the special part of the field, the prediction of performance properties of component-based systems, a number of approaches have been introduced so far, for an overview see [BGMO06]. They contribute to software engineering by predicting a non-functional property of a component-based system, thus adding to the predictability of the artefact to be designed and created.

However, the approaches have not become accepted in practical applications. One of the reasons for this might be that, although their accuracy has been tested by their authors in several case studies (type I validation), their applicability and usability has never been empirically validated (type II and III validations).

This thesis empirically validates the applicability of the Palladio performance prediction approach and compares it to the well-known SPE approach (type II validation).

Performance of Software Systems

Although hardware gets faster and more efficient each year, performance is nonetheless a critical factor when developing software systems. A major part of software projects fails to comply with performance requirements [Gla98], which leads to high costs or even project failure. Users are unwilling to accept long response times, and high response and processing time disturb system operation. The problem here is often not just to guarantee responsiveness for a fixed number of users, e.g. a group of test users, but to guarantee scalability, i.e. guarantee performance values also for increasing numbers of users. Even if the future load of a system can be estimated, systems might be tested with only a number of test users. They may perform well for the test load, but fail to meet performance criteria when used in production environments with a much higher number of users.

Two prominent examples for systems failing and causing high losses because of not complying with performance requirements are presented in [Koz04]: The automated baggage handling system at Denver airport and IBM's information system at the Olympic Games 1996 in Atlanta [SS01]. The initial problems with the baggage handling system caused the airport to open 16 month later than scheduled, almost $2 billion over budget and without an automated baggage system. Here, the system was planned to serve one terminal first, but later should serve all terminals of the airport [MK00]. The system was not able to cope with this increased demand, i.e. it was not scalable enough.

IBM's information system at the Olympics was tested with 150 users, however, 1000 user accessed it during the Olympic games, causing a system collapse. This failure caused the company high losses in reputation, not expressible in numbers [GR98]. Again, the system was not scalable to meet the timeliness requirements of the productive use.

In spite of these experiences, the performance of a software system is often not considered in the development process. A widespread attitude is to deal with performance problems when
they occur, i.e. after testing implemented parts of the system (fix-it-later approach, \[SW02\]).

Because performance problems are often based in the architecture of the system, their solving can become very costly at such a late point of time. Design decisions concerning the architecture have to be modified, which may lead to a new design and new implementation of major parts of the system.

To cope with this problem, performance prediction approaches have been proposed that predict the performance of a software system at early design stages and allow to identify performance problems in the architecture. Thus, their early use reduces the risk of expensive redesign phases later. This also adds to software engineering becoming an engineering discipline, as the prediction of characteristics of the designed artefact, both functional and non-functional, is crucial for an engineering discipline \[BFG^{+}04\].

**Component-Based Software Performance Prediction**

Since the beginning of the 80’s, the early analysis of non-functional properties, including performance, has been a topic of research. By analysing non-functional properties in an early stage of development, performance problems should be identified early and costly redesign and reimplementation should be avoided.

The term *Software Performance Engineering* (SPE) was coined by Connie U. Smith in 1981. She later defined it as a “systematic, quantitative approach to constructing software systems that meet performance objectives” \[SW02\] p.16], with being an “engineering approach to performance, avoiding the extremes of performance-driven development and ’fix-it-later’ ” \[SW02\] p.16].

SPE techniques are based on models describing the performance of the system to be developed. These models are attributed with certain performance values. In early stages of development, these values are based on estimation, in later phases existing implementation and prototypes can be used to get more precise values. Thus, Software Performance Engineering accompanies the whole development process.

As mentioned above, especially performance problems due to architectural flaws are problematic. This field gets more and more attention in recent times, many further approaches for predicting the performance at an early design level have been proposed \[BMDI04\]. An overview for performance prediction techniques at an architectural level is given in \[BMIS04\].

The prediction of performance is particularly supported by component-based systems, i.e. software systems assembled from software components. Software components are defined by Szyperski as follows:

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties” \[Szy98\] p.34].

Components have been initially introduced to support reuse \[BKR07b\]. However, their compositional structure and their contractually defined properties are also advantageous for performance prediction. Firstly, component-based systems may include reused and already implemented components at design time, which also means that their performance properties for
certain contexts are known or can be tested and thus do not have to be estimated. For components having to be newly developed or being hardly tested, the performance properties can still be estimated with SPE methods. Secondly, the use of contractually specified interfaces limits the degree of freedom for the later implementation already at design time. Finally, reusable component prediction models can be composed isomorphically to the software architecture, thereby lowering the effort for performance modelling.

However, component-based systems also pose challenges to performance prediction and quality prediction in general, so that classical techniques for performance analysis are unsuited for the performance prediction of generic software components [SKK+01].

An important aspect are the different contexts a component is deployed into. A static description of the quality of a component is impossible, because quality heavily depends on the context (platform, hardware, external calls, usage profile, etc.). Thus, the component has to be parametrised concerning its quality characteristics [BR06]. Additionally, the development process may be distributed among several developer roles, that all have incomplete knowledge on the components and system. This must also be considered by component-based prediction approaches. As a result, new performance prediction techniques have to be developed, specially made for the needs of component-based software engineering.

1.2 Contribution

The contribution of this thesis is twofold. Firstly, the applicability of the performance prediction technique Palladio is empirically evaluated and compared with the SPE approach from a user’s point of view. Secondly, an experiment design for this evaluation and comparison is presented.

The empirical evaluation focuses (a) on the applicability of the approaches and (b) on the identification of potential for improvement therein. For the applicability, the comprehensibility and the usability of both the approaches and the accompanying tools are studied (type II validation). Thus, in this thesis the human influences play an important role. This thesis does not evaluate the validity of the predictions themselves in terms of accuracy and precision (which would be a type I validation), that can be conducted as a case study and is not connected with the users applying the approaches.

To reach the above-stated goal, four main questions are posed:

- Question 1: What is the quality of the created performance prediction models?
- Question 2: What are the reasons for the model’s quality?¹
  - Question 2.1: Are the approaches comprehensible?
  - Question 2.2: Are the tools usable?
  - Question 2.3: What are further reasons?
- Question 3: What is the duration of predicting the performance?
- Question 4: What are the reasons for the duration?

¹Note that I understand “quality” to be ”similarity to a reference model” in this thesis, cf. section 3.2.2
CHAPTER 1. INTRODUCTION

1. Introduction

2. Performance Prediction

3. Research Method

4. Design and Conduction of the Experiment

5. Results

6. Conclusions and Outlook

Figure 1.1: Structure and line of reasoning of this thesis

In section 3.2.2, the questions are further explained and refined into metrics, using the Goal-Question-Metric approach [BCR94]. On this basis, an empirical experiment is designed. This design can also be applied to the empirical investigation of tool-implemented quality attribute prediction approaches in general. Thus, this research method forms the second contribution. The empirical analysis to apply the metrics to has the form of a controlled experiment.

The results were validated by studying the internal, construct and external validity of the experiment design, and reassessing them based on the outcome of the experiment.

1.3 Related Work

Related work can be found in the area of performance prediction, in particular for component-based systems, and in the area of empirical studies, in particular evaluating performance prediction approaches. I present related work where it thematically fits in this thesis.

Thus, related work for component-based performance prediction approaches is presented in section 2.2.2 for performance prediction approaches in general in section 2.3.

Related empirical studies are presented in section 3.1.2.

1.4 Structure of this Thesis

This thesis is divided into 6 chapters. Figure 1.1 gives an overview on both the structure of the chapters and the line of reasoning.

After this introduction, chapter 2 introduces foundations of performance predictions and the Palladio approach. In search of an approach to be compared to Palladio, several other component-based performance prediction approaches are presented and I argue why they are not suited for a comparison. After that, I present monolithic performance prediction approaches and describe
the chosen approach SPE. Chapter 2 concludes with a discussion of the comparability of the approaches.

Chapter 3 presents the research method. I first introduce foundations of empirical studies in software engineering and controlled experiments in particular. Then, I present related studies from the area of performance prediction approaches. After that, the research goal of this thesis is presented. Following the goal-driven Goal-Question-Metric approach [BCR94], questions and metrics are derived from the goal. Formal descriptions of the metrics are given.

Chapter 4 describes the experiment design and its conduction. It concludes with a discussion of the experiment design validity.

Chapter 5 presents the resulting data as collected with the before specified metrics. Additionally, the results are discussed and differences of both the approaches and the experimental tasks are pointed out. Finally, new findings on the validity of the experiments from the results are presented.

Finally, chapter 6 concludes and gives starting points for further research.
2 Performance Prediction

In this chapter, I first define performance and present foundations of performance prediction, in particular component-based performance predictions. Then, I describe the Palladio approach to be validated in this thesis. As I plan to compare Palladio to another performance prediction approach, I present other existing approaches for component-based performance prediction and argue why they are not suited for the comparison. Thus, I look at other, monolithic performance prediction approaches in section 2.3 I present some available approaches and argue why I chose SPE for the comparison. Finally, I discuss consequences for the comparison.

2.1 Theoretical Foundations

Firstly, I introduce my favoured definition of performance as defined by Smith and Williams:

"Performance is the degree to which a software system or component meets its objectives for timeliness" [SW02, p.4].

As two important dimensions of timeliness they name responsiveness and scalability. To continue with Smith and Williams,

"Responsiveness is the ability of a system to meet its objectives for response time or throughput" [SW02, p.4].

Thus, a responsive system responds fast enough to users or events, and can serve a sufficient number of users at the same time. For users, this means that they do not have to wait too long for the system during their work, even if other users are working with the system at the same time. For embedded systems such as airbag systems in vehicles, this means that the systems reacts within a – potentially very short – time period. In this explanation, the words "enough" and "too" suggest that responsiveness is relative and depends on the requirements for the system.

Next, Smith and Williams define scalability as follows:

"Scalability is the ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases” [SW02, p.5].

Thus, a scalable system will also be responsive if the demand to it increases, e.g. because more users use it or because the single tasks become more complex. With an increasing demand, the responsiveness should not or only slightly degrade. However, the demand to software systems always reaches a certain point at which processing resources are over-utilised and cannot cope with the demand, which results in exponential increase of response time [SW02, p.5]. Thus, a
CHAPTER 2. PERFORMANCE PREDICTION

scalable software system must either have enough reserves to meet a future higher demand or allow to be upgraded, e.g. by distributing it to multiple servers.

Note that this definition of performance does not include additional characteristics such as memory usage. However, this view fits the notion of performance that is used in the validated performance prediction approaches.

Performance prediction approaches aim at predicting performance metrics of software systems, which is desirable in several scenarios. Firstly, the performance of software systems can be analysed during design time, before actually implementing the system. Thus, high costs for the redesign of bad-performing architectures and systems can be avoided. Secondly, performance prediction can also answer questions that arise later in the software life-cycle. Scalability questions can be answered by predicting the performance of an existing system for a different – usually increased – use. The influence of deploying the software in other execution environments, e.g. to new servers, can also be analysed by using existing software models with new hardware models. Some performance prediction approaches support all of the scenarios mentioned above, others focus on specific areas. For example, the capacity planning approach as described in [MAD94] focuses on scalability questions for existing systems and do not support design time questions. Palladio, on the other hand, is designed to support all of the above scenarios [BKR07a]. Still, the overall process as well as the theoretical foundations, as explained below, remain similar.

Performance prediction involves creating performance models of the system and running analyses on them. Performance models describe the dynamics of software systems, i.e. the runtime behaviour, as performance is a run-time characteristic [BMIS04], as well as the other influencing factors on performance, such as the resource environment. More formal performance models such as queueing networks or stochastic Petri nets are hard to specify manually. Therefore, model-based approaches enable developers to specify software models in more abstract way, e.g. by describing the control flow and annotating performance values, and transform them into performance models for analyses. The abstraction can make the specification easier, additionally, developers might be more accustomed to more abstract models, such as sequence diagrams. Possibly, existing design documents can be reused and annotated.

Figure 2.1 (cf. [RH06]) shows the generic performance prediction process. Software design models, such as sequence diagrams or other control flow graphs, are annotated with performance metrics such as the CPU demand for a single computation. The result is an annotated software design model. Desirably automated with tool support, the annotated software design model is translated into a performance model (or analysis model) of the system, such as queueing networks or Petri nets. These models can be analysed using analytical methods or simulation. The analysis results, different performance metrics such as response time, utilisation or throughput, are fed back into the software design model to allow the user to draw conclusions.

For the performance models (or analysis models), many theoretical formalisms can be used, of which I present the most often used as presented in [BMIS04] in the following:

**Queueing networks** model each time-consuming resource of a system as a server with a prefixed queue. Figure 2.2(a) depicts this model. From single servers, networks can be built by connecting the servers as depicted in figure 2.2(b). Here, the left server represents a CPU, the right one a hard disk drive. Each job that arrives in the network is processed by
the CPU server. After that, the job either leaves the system or is processed by the disk and starts anew. Probabilities for the two different options are given. The shown network is an open queueing network, in which jobs arrive and leave the system. In closed queueing networks, a fixed number of users circulate in the network.

The complexity of a queueing network is determined by the type of queues used and the characteristics of the incoming jobs. For queues, the service time, the number of servers and the queueing policy is important. Service times can be constant (D), exponentially distributed (M), or arbitrary distributed (G). One or multiple servers can be connected to one queue. The queueing policy, i.e. scheduling policy, can be first-in-first-out (FIFO), processor sharing or prioritised, for example. The job are characterised by their arrival rate, which can be constant (D), exponentially distributed (M) or arbitrary distributed (G), and potentially different job classes with different characteristics, e.g. different service times at the servers [Bec08]. Classes of queueing networks are often defined by a triple X/Y/n, where X denotes the arrival distribution (e.g. M for exponential distribution), Y denotes the service time distribution, and n denotes the maximum number of servers per queue.

For a certain classes of queueing networks, analytical solutions exist. However, in these classes, rather restrictive assumptions were made. For many analytical solutions, exponential distributions for job arrivals and service times are assumed (M/M/n queueing models). This results in “memoryless” nets, as already passed time has no influence of the drawn sample, i.e. the processing time and arrival rates are stateless. For nets with arbitrary distributed arrival rates and service times (G/G/n), no known analytical solution exists [Bec08]. Here, simulation approaches are used to obtain performance metrics.

For M/M/n queueing networks, formulas for the relation of performance metrics are
known, i.e. if some performance metrics are given, the others can be calculated. For example, the mean residence time, throughput, and utilisation can be derived from the arrival rate (in an open model) and mean service time. However, the results are mean values for the performance metrics, and no information about their actual distribution or variance are known. For some requirements, e.g. quality-of-service contracts that specify that the response time should be lower than 3s for 70% of all requests, a distribution of the response time is needed to decide whether the requirement can be fulfilled.

For more information on queueing networks, see [BGdMT98].

**Stochastic Petri nets** (SPN) focus on synchronisation and concurrency within a software system. It differs from usual Petri nets in that transitions do not fire immediately, but only after a certain delay, usually specified using a distribution function. Thus, resource contention, mutual exclusion and priorities of tasks can be modelled.

The stochastic Petri nets combine functional and non-functional properties of software systems in one model. Additionally, properties of the concurrent behaviour such as deadlock freedom can be analysed. Many frameworks for SPN analysis exist, for an overview see [HM07]. However, for large systems, the number of states increases exponentially, so that the systems are no longer practically analysable [RH06, p.317]. For more information on stochastic Petri nets, see [BK96].

**Stochastic process algebras** (SPA) also focus on concurrent systems. In process algebras, the system is represented as a collection of processes, which communicate, interact and synchronise with each other. Processes can either represent atomic actions or be composed to form whole systems. Adding stochastic expressions, performance attributes can be represented. The behaviour of the processes can be described in detail, in contrast to queueing networks, that usually only contain probabilistic routes for jobs.

With stochastic process algebras, formal verification of the system specification, e.g. for deadlock freedom, is possible. As with SPNs, functional and non-functional properties of the software system are combined. However, the notation is fairly complex and hard to learn [RH06, p.317]. As for queueing models, no analytical solution is known for stochastic process algebras with arbitrary distributions [Bec08].

For more information on SPA for performance prediction see the survey in [HHK02].
To solve the performance models described in the above formalisms, analytical or simulation approaches can be used, depending on the class of the models.

**Analytical solutions:** The above-mentioned formalisms can be solved by converting and solving them as Markov chains (cf. [BGdMT98]), if sufficient assumptions are made. For special cases, e.g. the use of arbitrary distributions, the formalisms are no longer analytically solvable. In that cases, simulation has to be applied.

The advantage of analytical solutions, if applicable, is that they result in accurate performance values. At the same time, analytical solutions are faster than simulation, because the Markov chains can usually be solved efficiently and quickly [RH06, p.318]. However, the number of states in the Markov chains increases exponentially for growing complexity (state-space-explosion), so that at some point, no analytical solution is possible.

**Simulation:** With simulations, all of the above-mentioned formalisms can be analysed without exceptions. However, simulations only result in approximations of the accurate performance metric. The longer a simulation runs, the closer the approximation will be to the accurate value, but it is never certain how far the value is off. Confidence intervals using point estimators can be calculated for the results, but uncertainty remains. A further disadvantage is the computational effort: For significant results, simulations may have to run a long time [RH06, p.318].

**Hybrid approaches:** In hybrid approaches, analytical approaches and simulation are combined. Parts of the analysis is realised with analytical solutions, other parts are simulated. For example, intermediate result could be derived analytically and be fed into a simulation run.

### 2.2 Performance Prediction for Component-based Systems

Component-based software systems are particularly suited for performance prediction, because their composition may allow to predict the performance of a system based on the performance characteristics of the single building blocks [BKR07a]. However, component-based systems also bring in special challenges, because several independent developer roles are involved in the development process. Component developers have the knowledge on the control flow within components, system architects on the binding of components, and system deployers on the hardware and software the system is run on. As each role only has a restricted view on the system, no role has all the information needed for a performance prediction [BKR07a]. For example, the internals of the components should only be known to the component developer, firstly to hide complexity from the other roles, and secondly, if the component is used by third parties, to hide information.

Component-based performance prediction approaches have to handle these specific properties of component-based systems. They can use the compositionality of component-based systems and have to cope with the special development process.
Next to the development of new component-based systems, the scenarios introduced in section 2.1 such as scalability analyses, also apply to component-based performance prediction. Moreover, the context (i.e. environment) of the use of a component is even potentially unknown at design time, because a component is designed to be reused in several contexts, e.g. several systems. Overall, four factors of the context are identified in [BR06] to influence the performance of a component and thus of a component-based system: The actual implementation of the component, the hardware and software it is deployed on, the external services (e.g., of other components) it calls, and the usage (e.g., which services are called or which parameters are passed). Figure 2.3 depicts the four factors.

Performance prediction can be useful if any of the four influence factors changes: If new components should be designed and implemented, if the called external services of a component change because the assembly is changed, if the resource environment (i.e. the hardware the components are deployed on or the middleware) changes, and if the usage of components changes.

Again, as with traditional approaches, not all component-based performance prediction approaches take all scenarios into account (for an overview see [BGMO06]), but specialise in certain ones. Also, not all influence factors are taken into account explicitly.

### 2.2.1 The Palladio Approach

The Palladio approach [BKR07] is a component-based performance prediction approach that addresses the challenges described in the previous section. It supports the different roles in a component-based development process as shown on the left hand side in figure 2.4 namely component developer (who develops the components), system architect (who assembles components to form a system), system deployer (who allocates the components to hardware) and a newly introduced role, the domain expert (who provides information on how the system is...
used). Palladio is developed to allow performance predictions at design time, evaluating performance for the normal use of a component-based system (in contrast to worst-case performance prediction approaches for e.g. embedded systems).

Palladio meets the challenges of component-based systems – that (1) developer roles are separated and (2) the context of component use is unknown to developers – by introducing the Palladio component model (PCM). This component meta-model supports parametric dependencies and different views for different developer roles \[\text{BKR07b}\].

**Component developers** specify interfaces, components, and datatypes. They create parametric models called Resource Demanding Service Effect Specification (RDSEFF) \[\text{BKR07a}\], that model the control and data flow within components, also including calls to other services, and that are annotated with parametric resource demands. With the RDSEFF, the influence of the implementation is made explicit. The created models are stored in the PCM repository.

**Software architects** assemble components (respectively the models of the components) from the PCM repository by specifying connections between components. They do not know the control flow in a – potentially black box – component. By assembling components, they create a system and explicitly provide information on the second influence on component performance, namely the external services.

**System deployers** describe the available resource environment and specify on which nodes already assembled components of a system are deployed. Thus, they provide explicit information on the third influence, the resource environment of a single component.

**Domain experts** model the usage of the system by specifying the input parameters for requests to the system and the arrival behaviour of users. In doing so, they make the last influence, the user's behaviour, explicit.
The performance of a system can be predicted with a complete PCM instance consisting of a model having all four views, because every view specifies an influence on a component’s performance.

Next to addressing challenges of component-based systems, the PCM allows to specify most values, such as input parameters, resource demands, processing rates of the resource environment, and loop iteration in the control flow, as random variables. There are three reasons for this:

1. In doing so, it is allowed to not only specify constant parameters. For example, an integer parameter that can have four different values can be modelled.

2. Additionally, random variables reflect uncertainties during modelling, that can itself have several reasons: Firstly, the behaviour of the user usually cannot be determined for certain. The possible behaviours and used input parameters can only be described stochastically. Additionally, the processing rates of the underlying resource environment is influenced by environmental features such as garbage collection and middleware behaviour [RBK+07, p.28], that are too complex to be modelled in details and thus add further uncertainties.

3. Exponential distribution of time consumption cannot be assumed when black-box components are assembled, thus, random variables are needed when feeding the performance metrics of one bound and analysed component into another one.

In the PCM, random variables can be specified as discretisation of general distributions by specifying boxed approximations. An example with a discretised probability density function and the according graphical representation as a probability density is shown in figure 2.5. With a probability of 0.3, the value of the variable lies between 0.5 and 1, with a probability of 0.2, the value of the variable lies between 1 and 1.5, and with a probability of 0.5, the value of the variable lies between 1.5 and 2.

Additionally, it is possible to specify probability mass functions for integer or enumeration variables. Standard distributions such as the exponential distribution can be used with special keywords.

The response time predicted with the Palladio approach is again a distribution, which reflects both the uncertainties in the models (as random variables) as well as contention effects.

DoublePDF\{(0.5;0)(1.0;0.3)(1.5;0.2)(2.0;0.5)\}

Figure 2.5: Specification in the PCM (left) and graphical representation (right) of a random double variable
Analysis of the models

PCM instances can be used for different solutions and further design as shown in figure 2.4 on the right hand side. Firstly, the PCM instances can be solved analytical in special cases or using simulation. The analytical solution quickly delivers precise solutions, also supporting distributions of response times. In a first step of the solution, the specified parametric dependencies are solved using the information from usage profiles and resource environment. After that, the model with the solved dependencies is transformed into the performance model for the analytical solution, which is a restricted stochastic process algebra (SPA) based on regular expressions [KBH07]. Finally, the resulting response time distribution can be calculated as described by [FBH05].

Because the approach supports arbitrary distribution functions, other limitations are required. The analytical solution only handles single-user scenarios. Contention effects, parallelisms, and more advanced concepts like collection iterators and composite components are not supported. For queueing networks, these lead to arbitrary distributed arrivals at the servers, for which no analytical solution is known (cf. section 2.1). The mathematical and further assumptions made are presented in [KHB06].

For the simulation called SimuCom, a PCM instance is transformed into a simulation model based on queueing networks, more precisely, into Java code in the Desmo-J framework [DES07]. In this process, information from the usage models is transformed into workload drivers. The created threads execute according simulation code, drawing random numbers if needed for distribution functions. Generated probes collect data on performance metrics [Bec08]. The disadvantages of simulations are already introduced in section 2.1 and they also apply for the SimuCom: Results are only approximations and the simulation run may take a long time.

Both analysis methods make further assumptions on the models. The resource model is very simple, describing only some hardware resources and characterising these with mostly only a processing rate and a scheduling policy [RBK+07, p.92]. Further aspects such as context-switch costs for a CPU or dual core CPUs cannot be taken into account. Additionally, memory effects are neglected and no state of the system is considered. Furthermore, execution threads cannot be split from the control flow without a later join.

Next to solution models for analytical and simulated solutions, created using model-to-model transformations, a PCM instance can also be transformed into a Quality-of-Service (QoS) prototype, that contains generated time consumptions as specified in the PCM instance, as model-to-code transformation. The QoS prototype can be executed to measure the time consumption. Here, the effects of the real hardware can better be taken into account than with the abstraction used for the two previous solutions. However, the execution takes much longer than simulation and also requires the later hardware to be available, which both is expensive.

Finally, a PCM instance can be transformed into code skeletons that can be a basis for later implementations of the used components. The component developer only needs to add business logic to the method stubs generated from the RDSEFFs.

The right hand side of figure 2.4 gives an overview of the different uses of the PCM instance.

To include further information in the PCM instance, e.g. on the behaviour of the component container or the network, there exist further model-to-model transformations called completions.
to enhance the PCM instance. Firstly, the use of network resource is realised in this way. For each connection between two components that crosses resource container boundaries, a new component is introduced in a model-to-model transformation, that contains the needed logic to properly put demand on the network resource. It is also possible to further enhance this component and add information on the resource demands of the involved middleware. Secondly, a completion to represent dynamic lookups between the components using a broker is available. Here, components are added to the model for each connection between component, adding resource demand for the CPU to represent the computational effort for the lookup.

With this technique of adding information using model-to-model transformation, the component developer’s work is supposed to be eased, as he or she does not have to model influences of the middleware manually. The needed configuration of the system can be chosen in a menu, and is automatically added to the model using completions. Thus, the component developer can focus on modelling business logic and the corresponding resource demand of the components and the software architect does not have to specify additional information.

**Tool support**

The creation of a PCM instance and its analysis is supported by the Eclipse-based tool PCM Bench (version x.0.0.200707061844), making use of the Eclipse Modeling Framework (EMF, version 2.3.0) [Ecla] and the Eclipse Graphical Modeling Framework (GMF, version 1.0.100) [Eclb]. With the tool, an instance of the PCM can be created using the built-in EMF editor, that provide a tree view of EMF models, and graphical GMF editors, that provide specialised views for some parts of PCM. The component repositories with component and interface specification, the RDSEFF, the assembly of components to a system or to composite components, the allocation to hardware nodes, and the usage profile can be modelled graphically in specific GMF editors. The other needed models, i.e. the resource repository with the available resource types and the resource environment are modelled using the EMF editor.

Results of analysis or simulation are also shown in the PCM Bench. Response time distributions can be depicted as histograms or cumulated density functions, and utilisation in pie charts, all using JFreeChart version 1.0.5 [Gil07]. Additionally, simulation results can be fed into R reports (cf. [Dal03], available was version 2.5.0).

The screenshot in figure 2.6 shows the RDSEFF editor to specify a component’s control flow and resource demands in a parametrised way (upper main screen), as well as a view of the resulting distribution function for the specified system in form of a histogram (lower main screen). On the left hand side, an overview on the current project is given.

### 2.2.2 Other Component-based Prediction Approaches

To find an approach comparable to Palladio, I evaluated other component-based performance prediction approaches. Requirements for the comparability were that approaches allow to create performance models of the system under study at design time. The approaches may include an option to use measurements next to design documents, but approaches only focussing on the extrapolation of performance predictions from measurements are excluded from the list.
Figure 2.6: RDSEFF editor and response time histogram in the PCM Bench
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicable at design time</td>
<td>To find an experiment task applicable for both approaches, the approach must target design-time predictions. In particular, the compared approach must not depend on measurements alone.</td>
</tr>
<tr>
<td>Usable tools</td>
<td>In a study of the applicability, both methodology and tool are under study, thus, there must be usable tools to create the needed models. In particular, the potential transformation of design models into performance models and their analysis needs to be automated.</td>
</tr>
<tr>
<td>Normal-case analysis</td>
<td>In contrast to worst-case or best-case analyses, the compared approach must allow to study the system under normal, average conditions.</td>
</tr>
<tr>
<td>Component-based</td>
<td>As Palladio is a component-based approach, it is desirable to compare it to another component based approach.</td>
</tr>
<tr>
<td>Support of arbitrary distributions</td>
<td>As arbitrary distributions are an essential feature of Palladio, they should be included in the task description and in the required interpretation of the results.</td>
</tr>
</tbody>
</table>

Table 2.2: Requirements for an approach to be comparable to Palladio

Further requirements are usable tools, as both methodology and tool were under study in this thesis, as well as support of a normal-case analysis studying the system under normal, average conditions, and not worst-case scenarios. Additionally, the approach should support arbitrary distribution functions to allow corresponding experiment tasks that ask for their interpretation.

I summarise the requirements in the order of their importance in table 2.2. After assessing the approaches in detail, I summarise the findings in table 2.4.

A survey on component-based performance prediction approaches can be found in [BGMO06]. To limit the amount of approaches presented here, I filtered and only present approaches that are applicable at design time and are fairly matured in the following. There are only a few other component-based performance prediction approaches fitting these two criteria:

**KLAPER:** The KLAPER approach [GMS05] is specifically designed for predicting the performance of component-based architectures. It defines an intermediate language into which design-oriented notations can be transformed. Analysis models can be created from KLAPER instances using model transformations. Thereby, KLAPER reduces transformation complexity by treating components and resources in a unified way. However, there is no tool available to semi-automate the process, all model transformations need to be done manually, which is not feasible in an experiment and not realistic.

**LQN-Components:** In the LQN-Components approach [WMW03] [WW04], performance sub models for single components are created using layered queueing networks, i.e. queueing networks that consider both software and hardware contention. The models are parametrised, and can be assembled to system performance models, as components are assembled to form a system. Although a tool exists to create the system performance models from component sub models, there does not seem to be further coherent tool support to first create such sub models. Thus, the applicability of LQN-Components for this
CHAPTER 2. PERFORMANCE PREDICTION

Table 2.4: Evaluation of component-based approaches on requirements for comparability

<table>
<thead>
<tr>
<th>Approach</th>
<th>Applicable at design time</th>
<th>Usable tools</th>
<th>Normal-case analysis</th>
<th>Component-based</th>
<th>Arbitrary distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLAPER</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>LQN-Components</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>PACC</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>ROBOCOP</td>
<td>✓ ✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>CBSPE</td>
<td>✓ X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

None of the presented performance prediction approaches features arbitrary distributions, and most do not have a sufficient automated tool support, so that participants of an experiment are not able to apply it as they would in practical situations. ROBOCOP targets worst-case analyses instead of normal system operation. Thus, none of the approaches is applicable. As I expected
the least disadvantages from omitting the requirement that approaches should be component-based, I included monolithic approaches in the selection.

### 2.3 Monolithic Prediction Approaches

As the presented component-based performance prediction approaches were not applicable for a comparison, I looked at further approaches for performance prediction, that do not specifically target component-based systems. As the approaches do not make use of componentisation, I call them "monolithic" in this thesis, although "more general" would be another suitable term. Note, however, that they are not only to study monolithic single-server software systems, but may also target distributed, object oriented systems, for example.

To create performance models and conduct predictions, several monolithic approaches have been introduced, a survey can be found in [BMIS04]. Again, I filtered in advance and only considered approaches that allow to create a performance model of the system under study at design time. Of such, I present four fairly advanced approaches that also include tool support in the following. I applied the same requirements as presented in table 2.2, and present the results in table 2.6.

**SPE** with the SPE-ED tool [Smi90, SW02] is the first approach introduced in the field of software performance engineering, its creators also coined the term itself. It is a mature approach that is used in industry to predict the performance of systems. Additionally, former experiments in our research group attested the approach a good applicability in a replicated case study [Koz04].

**PRIMA-UML** [CM02] generates performance models from different UML models. Target models are a Software Model and a Machinery Model, which then again can be transformed into an Extended Queueing Network Model. The XPRIT tool accompanies the process of the PRIMA-UML approach. It transforms both sequence diagrams and use case diagrams created with the Poseidon UML tool and deployment diagrams created with ArgoUML into XPRIT models, and then creates execution graphs and queuing networks from them that can be analysed using tools such as SPE-ED. As ArgoUML is used, I expected similar problems as with CB-SPE (cf. section 2.2.2), and did not use the approach.

**CSM** with the PUMA tool set [WPS+05] is closely aligned with the UML-SPT profile ([Obj05]). The goal of this approach is to provide a common intermediate model (Core Scenario Model) different UML models can be translated in and different performance models such as queueing networks or stochastic Petri nets can be generated from. The PUMA tool set provides the means for the translation as a chain of different translators, e.g. a CSM2LQN tool. As the tool also depends on the annotation of UML models with profile information, it is questionable how they can be generated in a usable way [BKR07b].

**UML-PSI** [BM03a] is a simulation-based approach to performance modelling of software architectures specified in UML. Simulation model are derived from annotated UML software architectures, as specified with some UML diagrams, i.e., Use Case, Activity and
CHAPTER 2. PERFORMANCE PREDICTION

<table>
<thead>
<tr>
<th>Approach</th>
<th>Applicable at design time</th>
<th>Usable tools</th>
<th>Normal-case analysis</th>
<th>Component-based</th>
<th>Arbitrary distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PRIMA-UML</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CSM</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UML-PSI</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2.6: Evaluation of monolithic approaches on requirements for comparability

Deployment diagrams. A process-oriented simulation model automatically extracts information from the UML diagrams. Simulation provides performance results that are reported back into the UML diagrams. A prototype tool called UML-Ψ (UML Performance Simulator) implements the approach. Although the approach is fairly advanced, former experiments in our research group [Koz04] showed many problems when UML-PSI is applied by students in an experimental setting: The tool was instable and susceptible to wrong inputs. Thus, I did not use the approach again.

I decided to use the SPE approach with the SPE-ED tool, because it fits all requirements except that it is not component-based and does not feature arbitrary distributions (what none of the approaches does). Additionally, it is the only one also used in practical environments. The usability of SPE has already been shown in former work in our research group ([Koz04]).

In the following section, I present SPE in detail. In section 2.3.2, I argue why and under which conditions the two approaches Palladio and SPE are comparable in a study like this.

2.3.1 The SPE Approach

Software Performance Engineering (SPE) was the first approach introduced to predict the performance of software systems [Smi90], and already has been applied in industrial settings (several anonymised case studies are provided in [SW02]). It supports performance analysts, especially during early design stages, by providing methodologies, practices and a tool to predict and manage the performance of a system. Thus, in contrast to many of the previously introduced approaches, it does not only have an academic view on performance prediction, but does also include necessities for its practical application.

SPE was developed for monolithic systems, and later enhanced for distributed systems such as web applications. An SPE model specifies the performance-relevant control flow through the system. Additionally, means to analyse object-oriented systems with the help of UML sequence
diagrams were introduced [SW02]. Components are not directly supported, but their control flow can be described. The models are also hierarchical and thus offer limited composition possibilities.

The SPE process is ongoing during the software life cycle, but puts special emphasis on early phases. Proactive performance management is introduced, in which the performance of a system is predicted during design time, and not just tested later with implemented versions. Thereby, not the whole system is modelled. Performance analysts are required to identify key performance scenarios and model these. Later in the development process, models are refined using more detailed design and measurement results. Also, other performance modelling techniques might be used later in the development if more detailed models are needed [SW02, p.419]. Scalability questions can be analysed for existing systems. Overall, the principle is to keep models as simple as possible for a certain design stage. First, best case estimations are conducted, and only if these pass performance requirements, further detail is added.

Figure 2.7 shows the SPE workflow, that is "repeated throughout the SPE-inclusive development" [SW02, p.409]. First of all, the performance risk of the project needs to be assessed and the amount of effort for performance modelling needs to be defined. Then, performance critical use cases are identified. The motivation is, that only a small amount of functions of the system (<20%) use the major amount of time (>80%) [SW02, p.171]. These 20% need to be identified and analysed. The performance critical scenarios of the identified use cases are selected and performance objectives are set up, which is important to later know how to interpret the results. The performance models for the scenarios are constructed and annotated with software resource requirements. Computer resource requirements map the software resource requirements to actual processing devices of the hardware. Based on the evaluation of the models, the workflow continues with either finishing if the predicted performance is satisfactory, modifying the product concept (i.e. the software design) and later the models if possible and then continue, or revising the performance objectives if they proved infeasible. Next to creating the models and analysing them, they need to be ongoingly verified and validated. Further details of the workflow can be found in [SW02, p.407 et sqq.].

The workflow is repeated at several stages of the SPE process, each time with greater detail. Thus, the models are refined like the design itself is refined.
CHAPTER 2. PERFORMANCE PREDICTION

However, SPE does not only introduce the SPE process and workflow, but also practical considerations how to implement the process in an organisation and guidelines, principles and design patterns to design and develop high-performance software. This also includes performance management and information on how to collect data for the resource requirements estimation.

Each identified scenario is modelled using execution graphs (EGs), which describe the control flow of the applications using constructs such as loops, branches, parallel nodes, and synchronisation nodes. The meta-model for EGs and the underlying resource model is documented in [SLC+05]. The EG can be derived from UML sequence diagrams, for example, but it is not supposed to reflect every detail of execution, but to be an abstraction.

The EG is annotated with software resource requirements. The types of requirement can be chosen, e.g. ‘DB’ or simply ‘workunit’. Mean values are used, but not distributions. In an overhead matrix view, the software resource requirements are mapped to the hardware. For example, it can be specified that a software resource unit "DB" to describe the number of database accesses maps to 1000 CPU cycles, each taking 1ns to execute, and 1 hard disk drive access, taking 10ms.

For multiuser scenarios, mean values for the arrival rates can be specified, describing an underlying exponential distribution. Also, multiple scenarios can be allocated on different hardware facilities and be analysed together.

The results of analyses are response time (or residence time), time spent at one resource, and utilisation. For all metrics, only mean values are given as result.

Analysis of the models

The performance model of SPE is based on queueing networks. Different solutions exist, that are supposed to be used in the order they are presented below. The least-complex solution is used first, and only if a scenario meets its performance requirements, the next solution is conducted. In the following, I describe the four solutions as described in [Per03] and [SW98].

No Contention Solution: First, a scenario is analysed for the single-user user case. Here, the time needed for the single steps is calculated using overhead matrix and software resource requirements. After that, the time consumptions are added, potentially multiplied with loop repetition factors and weighted for branches. Results are mean residence time and mean time spent at the single resources.

Contention Solution: With the contention solution, single scenarios are analysed for multiple users arriving at the system in an either open or closed workload. Here, the queueing network is solved analytically using the known formulas (cf. section 2.1). Results are mean residence time, mean time spent at the single resources, and utilisation of the resources.

System Model Solution: Sets of scenarios running on one or several facilities concurrently are analysed with the system model solution. Here, a hybrid approach using both analytical "no contention" results of the single scenarios and simulation of the whole systems is applied. Resulting metrics are the mean response time for each scenario, the mean time spent at each hardware facility and the utilisation of each device.
Advanced System Model Solution: To support synchronisation nodes in the execution graphs correctly, the advanced system model solution was presented in [SW98]. Here, the simulation of the system model solution does also take the synchronisation behaviour of the single scenarios into account. The resulting metrics are the same as for the system model solution.

Like the Palladio approach, the SPE analyses only support limited resource modelling capabilities and neglect memory effects.

Tool support

SPE is supported by the SPE-ED tool (version 4) presented in [SW97]. It allows to create the models and solve them using the presented solutions. The modelling includes drawing execution graphs, specifying templates for software and computer resource requirements and facilities, annotating EGs with software resource requirements, and creating an overhead matrix to map the requirements. Furthermore, additional information for the solutions, such as arrival rates and simulation time, can be specified. The models can be solved with the four presented solutions.

Figure 2.8 shows an analysis view for a system with a single scenario, created using the system model solution. In the top-most view, the allocation of the scenario to hardware facilities and the simulated utilisation is shown. The two lower views show the residence times of parts of the EG. On the right hand side, the hierarchical composition of the system is symbolised: The more complex EG on the lower left side is actually part of the simpler EG on the lower right side, and expands the second node of the branch.

For this thesis, only an academic version of the SPE-ED tool with limited functionality was available. This version did not support correctly solving parallel nodes in the EG. If a step in the execution graph demanded hard disk drive access and a parallel step demanded CPU computation, the overall response time was calculated by adding the time consumed for the two steps, instead of using the larger value. Thus, parallel nodes were approximated by leaving out the estimated shorter branch in this thesis.

Additionally, the advanced system model solution was unavailable, thus correct synchronisation behaviour was not supported automatically. Here, a manual workaround approximated the synchronisation behaviour by (1) modelling all local system behaviour in one scenario, possibly splitting the behaviour for different use cases in the top-most EG, (2) analysing remote parts of the system separately and adding the response time as a delay in synchronisation nodes and (3) only using the contention solution for the resulting scenario. With this approach, the contention solution with its analytical solution can be used. However, it is not proven whether the results are the same and the modelling effort is higher.

2.3.2 Comparability of the Approaches

For this thesis, I decided to compare Palladio to SPE. However, as the SPE approach does not target component-based systems like Palladio does, I needed to find experimental tasks that fit
Figure 2.8: Analysis views of the SPE-ED tool
both techniques.

There are some aspects that can be realised with SPE, but not with Palladio:

- For servers in SPE, more than one resource of the same type can be modelled (e.g., a server can have 2 CPUs), and this fact is correctly reflected in the performance model. In Palladio, only a twice as fast CPU could be used, which of course results in wrong predictions, especially for the single user case. For a well-utilised system, the mean values might not deviate much, however, the distributions are wrong. Thus, the experimental task only used one resource per type and per server.

- With SPE, even asynchronous communication between scenarios is possible with the advanced system model. However, this type of solution was not available in my version of SPE, and thus cannot be used anyway.

- In SPE, it is possible to map one Software Resource Requirement, e.g. a database access unit, to several hardware resources, e.g. CPU and hard disk drive. This is not directly possible for Palladio, where each resource demand has to be separately defined in the software model (the RDSEFF), i.e., for one database access, CPU and hard disk drive demand need to be specified as two resource demands. However, as demands can be specified separately with Palladio, this does not restrict the experimental task itself, but only the presentation of software resource requirements.

- Usually, when applying SPE, only performance relevant scenarios of a software architecture are studied. However, for Palladio, usually the whole system is modelled, as the important aspects are unknown at design time. Thus, the experiment task also modelled a complete architecture and did not allow the participants to identify critical use-cases, which affects the time needed for applying SPE. However, as it is questionable whether critical scenarios can actually be identified at design time, this limitation is not very restrictive.

Other aspects can be realised with with Palladio, but not with SPE:

- SPE does not support a split of different roles to several developers. The views of SPE are closely related and do not have defined interfaces. Only the software model view (the execution graph) and the overhead matrix, mapping software resource requirements to hardware resource requirements, can be specified by two different people, if they agree on the available types of software resource requirements.

- SPE does not support arbitrary distributions. Thus, the task could not ask for the analysis of service-level agreements such as "70% of the requests must be answered within 3s". Additionally, the given input parameters must not be too skewed left to still have a meaningful mean value, which is used in SPE.

- With SPE, there is no model construct to model passive resources, such as semaphores. However, fix delays can be modelled. Thus, the delay at a passive resource because of multiple jobs arriving needed to be estimated manually.

- Parallel control flow is conceptually supported in SPE, but did not work with the available version (cf. section 2.3.1).
Thus, to be able to compare the approaches, the experimental task must not include several resource of one type (e.g., two CPUs), because it is unavailable in Palladio. It must not include asynchronous communication, because it is unavailable with both tool versions at hand. It must not split the development process on several people, as this is not supported by SPE. Thus, the modelling with Palladio was expected to be more time-consuming because of the extra effort for a splittable model, which did not result in benefits in this study.

Still, with these restrictions, reasonable experiment tasks were possible. Not all projects do have to include several resources of one type or asynchronous communication. Additionally, if only small example systems are analysed, the roles do not have to be split. Thus, Palladio can be compared to SPE in this thesis. However, the restrictions had to be considered when assessing the validity of the experiment (cf. section 4.3).
3 Research Method

As presented in the introduction, the empirical comparison of the two performance prediction techniques CB-SPE and Palladio was realised with a controlled experiment. Section 3.1 introduces different kinds of empirical research methods and describes why a controlled experiment was chosen.

Conducted without specific goals in mind, an experiment can lead to a large amount of data. To extract the relevant information after collecting the data can be hard, and it may be discovered that important information is missing, because its relevance was not recognised beforehand.

The goals of the experiment should be worked out in advance to be able to reduce the amount of data, eliminate irrelevant information, and collect all relevant information. A well known and successful goal-oriented procedure is the Goal-Question-Metric (GQM) approach by Basili et al. [BCR94]. Section 3.2 describes the GQM approach briefly and introduces the GQM plan for this thesis, containing questions and metrics to compare the performance prediction techniques.

3.1 Empirical Studies in Software Engineering

Although the discipline software engineering as a computer science branch is rooted in mathematics, it is nowadays understood as an engineering discipline [Pre01, p.30], as the name implies. In engineering, results are assessed based on their usefulness. This can often only be shown empirically, i.e., by experiencing and observing, and not deductively. Usually, software engineering processes are too complex for a thorough analysis. Thus, software engineering is not a deductive science, such as mathematics, in which new findings can be derived by conclusions and proofs.

To assess results, empirical evaluations must be conducted. Prechelt defines empirical evaluation as follows

"Empirical evaluation in the context of software engineering is the practical use and testing of a tool, method or model to understand and describe the actual characteristics of the artefact. By contrast, the speculative evaluation evaluation concludes the expected characteristics based on more or less plausible and mainly unexpressed assumptions by more or less stringently logical conclusions without empiricism” [Pre01] p.30] (translated by the author).

Prechelt names six different kinds of empirical studies: Case studies and benchmarking, field studies, controlled experiments, natural experiments, surveys, and meta-analyses [Pre01] p.35]. I briefly reproduce the six types in the following.
Case study and benchmarking: In case studies, methods or tools are assessed and described with applying it to a single concrete example. The example can be of toy-size or fairly complex. Case studies can be conducted in real surroundings or in laboratory settings. However, further influences on the outcome, such as the qualification of the users, are not systematically analysed or even excluded by design. The great advantage of case studies is their comparatively easy conduction. However, the interpretation of the results is difficult, as it is unclear to what observed results can be traced back. Benchmarks are a special form of case studies that are standardised and have a quantitative outcome.

Field study: In field studies, real software projects are observed. This can be useful if the subject of study cannot be simplified for laboratory settings or if a laboratory setting is too expensive. With field studies, much more complex situations can be analysed than with artificial settings. Additionally, the results apply to at least the studied real project. However, the generalisation to other projects and the investigation of the actual causes is problematic.

Controlled experiment: In a controlled experiment, only the factors being the subject of the empirical analysis are varied (treatment, experimental variables or independent variable), all factors that influence the outcome are controlled. Thus, the changes of the results (the dependent variables) can be identified as caused by the changes of the experimental variables. To control the influence of individual traits of a single user, many participants have to solve the same task, so that differences are balanced. However, this duplication makes controlled experiments costly. To reduce the effort, usually less complex tasks are studied.

Further advantages of a controlled experiment are the traceability of results to their causes and the good reproducibility. Because of this, when conducting an empirical study, the most reliable and convincing results are gained by conducting a controlled experiment.

Natural experiment: Natural experiments are a special case of controlled experiments and study often occurring tasks in software development that have to be carried out anyway. Thus, the participants of natural experiments are observed during their daily work. Only the way of solving the task, i.e. subject of study, is prescribed by the experimentators. Thus, the effort for natural experiments is low for the participants, but still very high for the experimentators.

Survey: With surveys, subjective information is collected from many people answering certain questions asked by the experimentators. However, only subjective opinions can be collected and consequently results have to be interpreted with care.

Meta-analysis: Meta-analyses combine the findings of several empirical studies on the same subject and thus consolidate the gained knowledge. With them, common results, missing aspects and also conflicts between the single studies can be detected, which leads to consolidated knowledge or ideas for further research. They are relatively inexpensive, however, they can only be conducted if enough comparable base studies are available.

For this thesis, I chose a controlled experiment as my research methods. With this form of empirical study, outcomes of the experiment can be traced back to the treatment, in this case the use of a specific performance prediction approach. Some other forms of empirical studies, such
as a natural experiment, a field study or a meta-analysis, are infeasible in the context of this thesis. The case study does not allow the tracing back of the outcome to results, and a survey on the one hand only results in subjective results, on the other hand requires practitioners applying the approaches.

In the following section, I describe the chosen empirical research method in more detail.

### 3.1.1 Controlled Experiment

The process of experimentation as described in [WRH+00, p.32] is shown in figure 3.1. The goal is to investigate the relationship between a cause and an effect, here the use of a certain performance prediction approach and the applicability. This relationship is depicted in the upper side in figure 3.1. This relationship is investigated by conducting a particular experiment, as depicted in the lower part in figure 3.1. A number of treatments (in this thesis the two performance prediction approaches) are applied in a certain experiment setting that allows control of other influencing factors. When conducting the experiment, a certain outcome is observed and thereby the relationship between treatment and outcome is investigated. If the experiment is properly set up, conclusions on the relationship between cause and effect are possible [WRH+00, p.32].

In a controlled experiment, it has to be ensured that apart from independent and dependent variables, all other influencing factors (the disturbance variables) are held constant. For empirically
comparing two performance prediction techniques, the experimental variable is the used prediction approach, whereas all other factors, such as the individual’s performance, must be constant. The observed outcome are the performance models created by the participants, judged with the metrics presented in section 3.2.

However, the effort to conduct a controlled experiment is considerably high [Pre01, p.45]. The claim to control all disturbance variables is hard to fulfil. Particularly if humans are participating in the experiment, which is most often the case in the context of software engineering, there are many influencing factors, such as the individual’s performance.

A large group of participants with preferably equal knowledge is a good way to minimize or at least identify the influence of uncontrollable variables connected to the individual’s performance. If participants are randomly assigned to the experiment groups and if the groups are large enough, it can be assumed that the individual’s influence is balanced in average [Pre01, p.53].

To keep the experimentation feasible, the tasks presented to the participants were fairly simple. The two concrete systems under study were the performance prediction for a Media Store application storing mp3 files and for a simple Web Server, both being artificial component-based systems. Still, except for their size, they were representative examples for business information systems. As two example systems were used instead of just one, the results here were better transferable to other architectures.

To achieve the best possible generalizability and to control the influencing factors as much as possible, I used guidelines and techniques for controlled experiments like presented in [Pre01]. When analysing the results of the experiment, I tried to identify the uncontrolled factors and interpret the results with this knowledge to assess the generalisability.

3.1.2 Related Empirical Studies

In general, there is little empirical research in software engineering [TLPH95, SDJ07]. Sjøberg et al. [SDJ07] analysed and meta-analysed scientific articles and found that only between 1.9% and 3% were controlled experiments, 1.6% were surveys, 2.3% - 12% were case studies (using varying definitions of what is a case study), and only very few action research. Overall, Tichy et al. [TLPH95] reported 17% empirical studies and Glass et al. [GVR02] reported 14% "evaluative" research.

Usually, empirical validation of performance prediction approaches study only a single performance prediction approach in the form of a case study and evaluate the accuracy of the approach by comparing it to measurements. There are exceptions, e.g. [BMDI04] and [BJHN04, BJH+05] (see below), in which different performance predictions approaches were compared. Still, in all known cases, the empirical validation was in the form of case studies.

[Koz04] gave an overview on empirical studies related to performance predictions up to the year 2004, that I reproduce in the following.

In [BMDI04], Balsamo et al. compared two performance prediction approaches that they developed in a case study for a real system. The older approach is based on process algebra and uses the Æmilia architectural description language. The performance models are derived
from UML sequence diagrams. As the approach suffered the state-space-explosion problem, they developed a second approach based on simulation. The simulation model is also derived from the UML descriptions and results in steady-state performance values. The approaches are compared using several criteria (performance model derivation, software model annotation, generality, performance indices, feedback, scalability, and integration). Their findings were that both approaches have advantages and disadvantages, but that they could be combined at affordable cost.

Smith et al. presented a case study in [SW93], in which they applied their SPE methodology to a temperature sensor system with some real-time properties and also showed how to evaluate performance characteristics of design alternatives. However, they did not compare the results of the predictions to measurements.

Next to the aforementioned ones, there are plenty of case studies in which researches show the validity of their own proposed approach, e.g. in [DRSS01] and [DAL04].

In [Koz04] itself, three performance prediction approaches were compared in a replicated case study with student participants, similar to this thesis. The performance prediction approaches under study were SPE [SW02], umlPSI [Mar04] and Capacity Planning [MAD94]. The study attested SPE a good applicability for early design time predictions, found that Capacity Planning is rather fitting for the analysis of existing systems than for early design time predictions, and uncovered numerous problems with the umlPSI tool, although the approach itself was very promising.

After 2004, Bacigalupo et al. have compared their performance prediction technique HYDRA [BTJN03] to a performance prediction approach using layered queueing networks in two case studies [BJHN04] [BJH05]. HYDRA extrapolates from historical performance data for different classes of workload and is used for Grid-applications. In both cases, benchmarks were used to generate loads for measurements and to validate the predicted values.

In 2005, I compared an earlier version of the Palladio approach as presented in [FB04] [FBH05], with only a rather immature tool support, to CB-SPE [BM03b] [BM04] in a smaller replicated case study [Mar05]. The results attested the CB-SPE approach conceptually a good applicability, although many problems with the tools occurred. The Palladio approach had less good results. There were many problems with the specification of the distribution functions in the immature tool. However, as the Palladio approach and tool evolved since then, a further study is useful.

After describing foundations of empirical studies and related work in this section, I introduce the research goal of this study and derive questions and metrics in the next section.

### 3.2 Goal-Question-Metric Plan

The primary principle of the GQM approach [BCR94] is that the collection of empirical data should be goal-oriented, i.e. focus on and pursue a defined goal. The first advantage of this principle is that, having the goal in mind, it is easier to choose useful and relevant data. This is supported by the top-down approach of GQM: On the basis of the goals, questions are defined which make the goal operational and further lead to metrics. The second advantage of the GQM
approach comes with the interpretation of results: In a bottom-up approach, the collected data is interpreted based on the questions and finally based on the goals [DHL96]. The goals, questions and metrics together form the GQM plan.

There are several prerequisites for a successful use of the GQM approach [DHL96]:

1. The goal must specify in great detail what is to be analysed.
2. Metrics have to be derived in a top-down fashion based on goals and questions.
3. The choice of metrics must be explicitly documented. The GQM questions embody this rationale of how the metrics are derived from the goals.
4. The collected data must be interpreted in a bottom up approach based on the GQM questions and goals.
5. The people whose viewpoint is used in the GQM goal have to be deeply involved in the definition and the interpretation of the goal.

Prerequisite 1 to 4 were be taken into account in this GQM plan, and are explicitly named where fulfilled. Prerequisite 5, however, relates to application of the GQM approach in practical surroundings, e.g. in software development. In research, the participants of an experiment are almost never involved in the design of the GQM plan. Thus, this prerequisite was invalid in this thesis.

The detailed GQM plan is also interlinked with the actual experiment design. For example, only an experiment design with multiple tasks can be used to compare two approaches in respect of the features of the task they are applied to. With only one task, the outcome may be caused by either the independent variable or by characteristics of the task. Furthermore, to assess a characteristic of the approach on average over several participants and/or tasks, the calculation of the average cannot be specified without knowing the details of the experiment design. Constraints on the experiment design such as organisational and financial ones may also affect the questions and metrics in reality. Still, the goals, questions, and metrics should define the experiment design, and not the other way around.

Some requirements for the experiment design have already been presented in section 3.1.1. To keep the separation of GQM plan and experiment design, I present the experiment design in its own chapter 4. However, there are also implications for the experiment directly from the GQM plan that are mentioned in this section.

3.2.1 Goal of the Experiment

A GQM goal specifies the purpose of measurement, the object to be measured, the issue to be measured, and the viewpoint from which the measure is taken [BCR94]. Naming all these parts of the goal fulfils prerequisite 1. Here, the GQM goal was to

empirically validate the applicability of the performance prediction approach Palladio from a user’s point of view.
CHAPTER 3. RESEARCH METHOD

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Empirically validate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue</td>
<td>the applicability</td>
</tr>
<tr>
<td>Object</td>
<td>of the performance prediction approach Palladio</td>
</tr>
<tr>
<td>Viewpoint</td>
<td>from a user’s point of view</td>
</tr>
</tbody>
</table>

Table 3.1: Research Goal

Note that the term ‘user’ stands for the different roles involved in the performance prediction. For the Palladio approach, these are all roles involved in the process of developing component-based systems, namely component developer, system assembler, component deployer and QoS analyst, as presented in section 2.2.1. Of course, it does not mean the end user of the developed component-based system.

To assess the applicability, a evaluation scheme to judge the outcome is needed. One possibility is to set up a set of requirements to check against, a second is to compare the applicability to another performance prediction approach as a reference. As the setting-up of requirements is highly subjective, I compared Palladio to another performance prediction approach. The SPE approach was chosen (see section 2.3) as it is a commonly used technique which is also practically used in industry [SOB01]. Thus, it is a suitable reference for comparison.

Consequently, the mean to achieve the goal is to

empirically compare the applicability of the Palladio approach and the SPE approach from a user’s point of view.

Note that the applicability was under study and not the ability of the approaches to yield accurate and precise predictions for real software systems. Thus, the emphasis lies on how participants are able to create the required models. The resulting predictions are not compared to measurements of implementations.

To assess the applicability of Palladio, I paid attention to its characteristics as introduced in section 2.2.1. Thus, some metrics relate to Palladio’s parametrisation and the specification of distribution functions. The target scenario and capabilities of the Palladio approach were also considered in the experiment design: The experiment tasks (described in section 4.2.2) are about a component-based system with one design alternative allowing automated model completion (cf. section 2.2.1). However, development roles and the interpretation of distributions were not under study, because they cannot be handled with SPE and thus could not be integrated into the experiment design (cf. section 2.3.2).

3.2.2 Questions and Derived Metrics

Based on the GQM goal, I derived four questions to be answered with this study. For the applicability of the performance prediction models under study, two important factors are (1) the quality of the created models and (2) the duration of a prediction. Only if created models have a sufficient quality and can be created in a certain amount of time, the approaches are applicable. For both characteristics, I asked for (1) their degree in the experiment and (2) for the reasons of the observed outcome, to be able to draw conclusions. For all four aspects, both
the methodology and the tools are under study. As I expected the reasons for the observed quality to be the comprehensibility of the approaches and the usability of the tools, I asked sub-questions specifically asking for these reasons and one sub-question also asking for further reasons, allowing further insight.

The following list presents the resulting four questions.

- **Question 1**: What is the quality of the created performance prediction models?
- **Question 2**: What are the reasons for the model’s quality?
  - Question 2.1: Are the approaches comprehensible?
  - Question 2.2: Are the tools usable?
  - Question 2.3: What are further reasons?
- **Question 3**: What is the duration of predicting the performance?
- **Question 4**: What are the reasons for the duration?

In the following, the four questions are presented in detail with hypotheses and 21 metrics, thus fulfilling prerequisite 2. A detailed rationale is given for each question and a hypothesis is stated, thus fulfilling prerequisite 3. After the rationale, first an overview of the metrics of the questions is given, followed by a detailed description of each metric, that includes a formal description in the case of quantitative metrics. Questions are stated independent from the actual experiment tasks, to enable future experimentators to reuse them with other experimental set-ups. As argued above, some metrics take the experiment design into account, but they can be replaced or omitted.

For the following discussion, I introduce the following variables:

- **Palladio approach**: $Pal$
- **SPE-ED approach**: $SPE$
- **Set of approaches to be compared**: $A = \{Pal, SPE\}$
- **Set of systems to be analysed is the Media Store and the Web Server**: $S = \{MS, WS\}$
- **Set of variants to be analysed for each system $s \in S$**: $V^s = \{v^s_1, v^s_2, v^s_3, v^s_4, v^s_5\}$
- **Set of usage profiles to be analysed for each variant**: $UP = \{UP_1, UP_2\}$
- **Set of participants applying approach $a \in A$ for system $s \in S$**: $P_{a,s}$
- **Arithmetic mean of a set of real values**: $avg$

Table 3.2 on page 37 gives an overview of all derived questions and metrics. Altogether, 4 questions, one of them divided into 4 sub-questions, and 21 metrics will be used to analyse the experiments results.
Table 3.2: Summary GQM Questions and Metrics

<table>
<thead>
<tr>
<th>Question 1</th>
<th>What is the quality of the created performance prediction models?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M1.1</td>
<td>Relative deviation of predicted mean response times of the participants and of the reference model.</td>
</tr>
<tr>
<td>Metric M1.2</td>
<td>Passed K-S Test ratio of predicted response time distribution and reference</td>
</tr>
<tr>
<td>Metric M1.3</td>
<td>Percentage of correct design decisions.</td>
</tr>
<tr>
<td>Metric M1.4</td>
<td>Permutations in design decision rankings, recognising clusters.</td>
</tr>
<tr>
<td>Hypothesis 1</td>
<td>With both approaches, the created models are similar to the reference model.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 2</th>
<th>What are the reasons for the model’s quality?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M2.1</td>
<td>Problems when creating the models and classification</td>
</tr>
<tr>
<td>Hypothesis 2</td>
<td>Some potential problems arise from a lack of understanding and tool difficulty.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 2.1</th>
<th>Are the approaches comprehensible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M2.2</td>
<td>Number of times of rejection before acceptance level is reached.</td>
</tr>
<tr>
<td>Metric M2.3</td>
<td>Number of interpretation problems</td>
</tr>
<tr>
<td>Metric M2.4</td>
<td>Subjective evaluation of comprehensibility by the participants.</td>
</tr>
<tr>
<td>Metric M2.5</td>
<td>Subjective evaluation of distribution functions by participants.</td>
</tr>
<tr>
<td>Metric M2.6</td>
<td>Subjective evaluation of parametrisation by participants.</td>
</tr>
<tr>
<td>Question 2.2</td>
<td>Are the tools usable?</td>
</tr>
<tr>
<td>Metric M2.7</td>
<td>Subjective evaluation of the tool usability by participants.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 2.3</th>
<th>What are further reasons?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M2.8</td>
<td>Analysis of explanations in questionnaire to find additional influences.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 3</th>
<th>What is the duration of predicting the performance?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M3.1</td>
<td>Average duration of a prediction</td>
</tr>
<tr>
<td>Metric M3.2</td>
<td>Time needed to solve preparatory exercises</td>
</tr>
<tr>
<td>Metric M3.3</td>
<td>Subjective evaluation by participants on needed time and effort to learn the approaches</td>
</tr>
<tr>
<td>Hypothesis 3</td>
<td>The duration for a Palladio prediction is 1.5 times higher as the duration for an SPE prediction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 4</th>
<th>What are the reasons for the duration?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric M4.1</td>
<td>Duration of the single steps</td>
</tr>
<tr>
<td>Metric M4.2</td>
<td>Breakdown of the duration to activities</td>
</tr>
<tr>
<td>Metric M4.3</td>
<td>Subjective evaluation by participants on reasons for the needed time</td>
</tr>
<tr>
<td>Hypothesis 4</td>
<td>The most time-consuming activity is the modelling.</td>
</tr>
</tbody>
</table>
1. What is the quality of the created performance prediction models?

**Rationale** A performance prediction is only successful if the created model of the system under study reflects the performance properties of the system well.

As this thesis validated the applicability of the approaches (type II validation) and not the accuracy and precision of the predictions (type I validation, cf. section 1.2), it was not the question whether the predicted response times were realistic, i.e. could be found for real implementations. To exclude the influence of type I aspects, the results were compared to a reference model that had been carefully created including all given information, and not to measurements of an implementation. As this reference model represented what is needed in the approach to yield good predictions, it was tested whether a similar model can be created by the participants. Thus, in the following, quality of of the models is defined to be the similarity to the reference model. In doing so, I measured the applicability in terms of (1) how well the participants understand the approaches, (2) how well they are able to realise their knowledge, and (3) how usable the given tools are.

Still, the similarity to the reference model is not straightforward to measure. One option was to compare the created model to the reference model based on a metric to compare models. However, such a metric was again difficult to create. Is it structural similarity or rather similarity of annotations that make models similar? Because this was unclear, I came back to the outcome of a prediction and assessed a model the better, the closer the predictions are to the predictions of reference model. Thus, the similarity important for performance predictions was taken into account. Note that the participants were given all needed information on the system, and did not have to estimate any performance annotations. If I had included estimations as a further influence factor, problems with the applicability could not have been distinguished from mere estimation mistakes, thus the outcome could not be traced back to the independent variable only.

Still, the goals that should be achieved with the prediction determine whether a performance model is good, or rather whether it is good *enough*. For critical applications, either safety critical or business critical, a more accurate prediction of the response time might be needed, considering every detail that somewhat affects the performance. In other cases, for example when deciding for a design alternative out of a set of possibilities, it might be important to get the relation of the response times, but their exact values are of less interest. Thus, for assessing the similarity of the models, not only the absolute predicted values were taken into account, but also the ranking of design decisions.

However, changes in the models do not necessarily result from mistakes of the user. Participants might choose to deliberately model the system differently than suggested in the task description, e.g. to add annotation that seem realistic to them and that might even make the prediction more realistic in their opinion. For example, participants might chose to add CPU demand to an RDSEFF to reflect garbage collection or similar effects. Such changes needed to be identified and handled individually. They were removed to be able to compare the model to the reference.

My hypothesis 1 was that with both approaches, the created models are similar to the reference model. Participants should be able to create similar models after a intensive training. However, no quantitative measure was given, as there are no known bases for it.
Overview of the metrics  First, a performance model should deliver values that are similar to the reference performance model when analysed. Here, the predicted response time was an important performance metric. To assess for which approach the predicted response time was closer to the predicted response time of the reference performance model, the relative deviation between predicted and reference mean response times was the first metric \(\text{M1.1}\).

For Palladio, the predicted response time was also given as a distribution function. To assess the quality of the predicted distribution of the response time, metric \(\text{M1.2}\) compared the predicted distribution of response time with the corresponding distribution of the reference model.

As mentioned above, the absolute predicted response time is not always important for a performance prediction. To assess different options when designing or changing a system, the relation of the respective response times is of interest. Therefore, it was measured how many participants identified the best design option in respect of its response time by stating metric \(\text{M1.3}\) as the percentage of correct design decisions.

Next to identifying the best design options, all options were also ranked. To assess how well the response time of different alternatives could be predicted, the ranking of design decisions done by the participants was compared to the ranking of the response times of the reference solution in metric \(\text{M1.4}\).

The following enumeration summarises the metrics for question 1:

- **M1.1**: Relative deviation of predicted mean response times of the participants and of the reference model
- **M1.2**: Passed K-S Test ratio of predicted response time distribution and reference (Palladio only)
- **M1.3**: Percentage of correct design decisions
- **M1.4**: Permutations in design decision rankings, recognising clusters

Detailed description of the metrics

**Metric \(\text{M1.1}\): Relative deviation of predicted mean response times of the participants and of the reference model.**  For metric \(\text{M1.1}\) the absolute deviation was favoured over the standard deviation, as it is superior for small sample sizes and if extreme values are expected [Sac97] p.335].

I first measured the deviation from the mean response time predicted for the reference model separately for each variant \(v \in V^s\) of each system \(s \in S\) and each approach \(a \in A\), calculating the deviation between the predicted response time for each participants and the reference response time, and then averaging the deviation over all participants.

Let \(\text{predMeanResp}_{v,u,p}\) be the mean response time that participant \(p \in P_{a,s}\) predicted for system \(s \in S\), variant \(v \in V^s\), and usage profile \(u \in UP\) and let \(\text{refMeanResp}_{v,u,a}\) be the mean response time that was predicted with the reference performance model for system \(s \in S\), variant \(v \in V^s\), and usage profile \(u \in UP\). The absolute deviation between both values was averaged:
\[
\text{absDevMeanResp}_{v,u,a} = \text{avg}\{\text{predMeanResp}_{v,u,p} - \text{refMeanResp}_{v,u,a} \mid s \in S, v \in V^s, p \in P_{a,s}\}
\]

In order to compare the deviation for the different variants, the proportion of the deviation to the reference response time was calculated:

\[
\text{propDevMeanResp}_{v,u,a} = \frac{\text{absDevMeanResp}_{v,u,a}}{\text{refMeanResp}_{v,u,a}}
\]

In doing so, the influence of the specific task on the deviation could be analysed. Additionally, the metric was measured over all systems, variants, and usage profiles to directly compare the approaches:

Metric [M1.1]  
\[
\text{propDevMeanResp}_a = \text{avg}\{\text{propDevMeanResp}_{v,u,a} \mid s \in S, v \in V^s, u \in UP\}
\]

Metric [M1.2] Passed K-S Test ratio of predicted response time distribution and reference (Palladio only)  
The Palladio approach also takes into account, that the response time of a system is a distribution function. Not all calls to the system have the same response time, due to various reasons (e.g. changing parameters, speed of the underlying hardware, contention effects etc, cf. section 2.2.1). To assess the quality of the predicted distribution of the response time, metric [M1.2] compared the predicted distribution of the response time with the corresponding distribution of the reference model. This metric could only be measured for the Palladio results.

The result of a Palladio prediction by simulation is not a continuous distribution function, but a set of many drawn samples. To compare the predicted distributions, statistical tests to assess whether they resulted from the same underlying distribution can be used. The hypothesis to be tested was whether the two predicted distributions (by the participants and the reference) resulted from the same underlying distribution function. Several methods exist to compare distributions, each having advantages and limitations. The Kolmogorov-Smirnov (K-S) test does not depend on testing against a specific underlying distribution function like the normal distribution, but may be applied to all kinds of distribution functions. The null hypothesis is that two sample distributions result from the same underlying distribution function. A calculation can be done with the R tool [Dal03] as described in [MTW03].

Another approach often used is the \(\chi^2\) goodness of fit test [Pea00]. It tests whether two sample distributions are independent of each other, i.e. the null hypothesis is the opposite of the null hypothesis of the K-S test, which must be considered when interpreting the results. However, its R implementation is computationally complex and in my tests it took several seconds for the comparison of two samples having 2500 values each. Additionally, out-of-memory errors occurred for larger samples. As Palladio prediction results contained even more data for the experimental task, the \(\chi^2\) test was not used.
CHAPTER 3. RESEARCH METHOD

Applying the K-S test results into a test value and a p-value. The p-value is the probability of the null hypothesis being true. If the p-value is lower than a significance level, the null hypothesis is rejected. Here, I used a significance level of \( p = 0.05 \), i.e. I rejected the null hypothesis if the probability of the null hypothesis being true is less than 5%.

The K-S test was applied to compare each predicted distribution to the respective sample distribution. Let \( df_{p,v,u} \) be the distribution that participant \( p \in P_{pal,s} \) predicted for variant \( v \in V^s \) and usage profile \( u \in UP \) and let \( refdf_{v,u} \) be the distribution that was predicted with the reference model. The null hypothesis was that \( df_{p,v,u} \) and \( refdf_{v,u} \) resulted from the same underlying distribution function. I rejected this null hypothesis if the p-value of the K-S test was smaller than 0.05. Thus, a distribution passed the test if the p-value was greater than 0.05 or equal. Now, it could be analysed how many of these distribution functions passed the test:

\[
PassRatio_{v,u,Pal} = \frac{|\{df_{p,v,u} | df_{p,v,u} \text{ passes the test, } p \in P_{pal,s}\}|}{|\{df_{p,v,u} | p \in P_{pal,s}\}|}
\]

Additionally, it was analysed how many distribution functions passed the test overall by calculating the ratio over all variants \( v \in V^s \), systems \( s \in S \) and usage profiles \( u \in UP \).

**Metric M1.2**

\[
PassRatio_{pal} = \frac{|\{df_{p,v,u} | df_{p,v,u} \text{ passes the test, } s \in S, v \in V^s, u \in UP, p \in P_{pal,s}\}|}{|\{df_{p,v,u} | s \in S, v \in V^s, u \in UP, p \in P_{pal,s}\}|}
\]

**Metric M1.3 Percentage of correct design decisions**

Metric M1.3 was the percentage of correctly identified best design decision, and was defined as follows. Let \( DD_{s,u,a} \) be the set of design decisions of the participants for a system \( s \in S \), a usage profile \( u \in UP \) and the approach \( a \in A \). Then, the percentage of correct design decisions was:

\[
perc_{s,u,a} = \frac{|\{d | d \in DD_{s,u,a}, d \text{ is correct}\}|}{|DD_{s,u,a}|}
\]

For all systems and usage profiles, I obtained:

\[
Metric M1.3 \quad perc_a = \frac{|\{d | d \in \bigcup_{s \in S, u \in UP} DD_{s,u,a}, d \text{ is correct}\}|}{|\bigcup_{s \in S, u \in UP} DD_{s,u,a}|}
\]

**Metric M1.4 Permutations in design decision rankings, recognising clusters**

For metric M1.4, which measured the permutation of the rankings, it was taken into account that there might be options that were very similar in respect to response time, and others that differed greatly. Thus, the permutations in design decision ranking was looked at, recognising clusters of similar response times (metric M1.4). This metric was measured separately for each system under study.
Counting the number of correct ranks was not enough. There is a difference between the quality with a permutation of two neighbouring ranks and the permutation of the first and last rank. Thus, the difference between the right rank and the actual rank must be taken into consideration. Here, a metric proposed in [FKB+05] was used with slight changes. A ranking of the predicted response times of the reference performance model for each system $s \in S$, usage profile $u \in UP$ and approach $a \in A$ was captured in the mapping

$$\text{PosReference}_{s,u,a} : V^s \to 1, \ldots, |V^s|$$

with $\text{PosReference}_{s,u,a}(\text{variantB}) < \text{PosReference}_{s,u,a}(\text{variantC})$, if the mean response time of variant B was smaller than the response time of variant C. The ranking of each participant $p \in P_{s,s}$ was captured in a similar mapping $\text{PosPred}_{u,p}$. To recognise similar variants in respect to their response time, the variants were clustered. The mapping

$$\text{class}_{s,u,a} : 1, \ldots, |V^s| \to 1, \ldots, |\text{classes}|$$

is monotonically increasing and maps each position of the ranking of the reference performance model to the class of the associated variant. Thus, the permutation can be defined as

$$\text{perm}_{u,p}(\text{variant}) = |\text{class}(\text{PosPred}_{u,p}(\text{variant})) - \text{class}(\text{PosReference}_{s,u,a}(\text{variant}))|$$

For each participants’ predicted ranking, I obtained a permutation score of

$$\text{perm}_{s,u,p} = \sum_{v \in V^s} \text{perm}_{u,p}(v)$$

To make this metric clear, I give two examples, leaving the indices away: Let us assume the reference ranking of the alternatives A1 to A4 is A1, A2, A3, A4 in that order. A3 and A4 have very similar response times and form a cluster. Assume a ranking $r_1$ to be A4, A2, A1, A3. Now I have $\text{class}(r_{ref}(A4)) = 3$ and $\text{class}(r_1(A4)) = 1$, a difference of $2 = \text{perm}(A4)$. Overall, $r_1$ has an permutation of $\text{perm} = 2 + 0 + 2 + 0 = 4$.

Thus, the permutations of a whole ranking can be given as the sum $\text{perm}_{s,u,p}$. However, what does a permutation of 8 mean? Of course, this depends on the number of possible permutations. If the ranking has 4 ranks, 8 is the maximum permutation (as defined above) that can be reached. However, if the ranking has 100 ranks, 8 is a low permutation that might have come out of the interchange of rank 52 and 60, for example, when all other ranks are correct.

To be able to compare two systems for which a different amount of classes have been defined, I needed a metric that was independent of the actual numbers of ranks and classes, i.e. normalised. The normalisation could be done by dividing by the maximum permutation for the given number of classes and ranks. Let $i$ be a fictitious participant who ranked the inverse of the correct ranking. Then, the maximum permutation can be determined by calculating $\text{perm}_{s,u,i}$. Now, the proportion of permutations was given with:
\[ \text{propPerm}_{s,u,p} = \frac{\text{perm}_{s,u,p}}{\text{perm}_{s,u,i}} \]

This proportion was averaged over all participants using one approach and all usage profiles to get a metric for the comparison of the approaches:

\[
\text{Metric } M1.4: \quad \text{propPerm}_a = \text{avg}\left(\{\text{propPerm}_{s,u,p} \mid s \in S, p \in P_{a,s}, u \in UP\}\right)
\]

2. What are the reasons for the model’s quality?

**Rationale** The previous questions asked for the prediction accuracy compared to a reference model for each of the approaches. To be able to draw conclusions from the measured quality metrics, I needed to ask for the reasons. This enabled the distinction of certain aspects of the applicability of the approaches.

Several factors might influence the quality of a prediction. First of all, the participants needed to understand the approaches and their various concepts. Additionally, the tools must 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 interpretation might lead to false conclusions. This depended on the results the approach gave as well as on visualisation of results in the tool.

There may as well be factors that were not foreseen in the design of this GQM plan. To nonetheless be able to capture such factors, qualitative questions were asked.

Accordingly, my hypothesis 2 was that potential problems arise from a lack of understanding and tool difficulties.

To further structure this question, sub-questions were introduced for each of the influencing factors asking for comprehensibility, tool support, and possible other reasons:

- Question 2.1. Are the approaches comprehensible?
- Question 2.2. Are the tools usable?
- Question 2.3. What are further reasons?

Before these sub-questions are elaborated in detail, a base metric is defined.

**Overview of the metrics** A base for most metrics of this question was the number of problems and errors in specific aspects when creating the models. Thus, this metric was measured first to be usable in all following metrics. Not only errors were interesting. Because (1) the participants had the possibility to ask questions during the experiment and (2) their results were tested before being accepted, some problems might be caught before resulting in an error in the final model. Thus, not only actual errors, but also documented problems during the experiment were considered in this metric.

M2.1. Problems when creating the models and classification
Detailed description of metric M2.1: Problems when creating the models and classification

Problems might have resulted from several aspects: A participant could have a problem with the approach itself (i.e. comprehension), a problem with the implementation of the tool or a problem with the task of the experiment. Again, it could not be foreseen whether there were other problem areas. An additional class of error sources could be tight time constraints or lacking motivation. However, this class of errors could not be readily distinguished from the others. The experimental set-up needed to ensure that the participants were not stressed and properly motivated. Excluding the error sources motivation and stress, the other kinds of errors could be distinguished by analysing a specific error.

Not all errors and problems led to a decreasing quality, hence only performance-critical problems were looked at. Problems could have different severities. Three examples were presented in the following. A major problem would be not knowing how to use loops, although the task required it. An intermediate problem would be omitting a value that influences the performance, but not crucially. A minor problem would be to forget a single small step, that overall did not or only barely influenced the performance. The classification of the severity of errors was arbitrary and did not found on fix criteria. Still, with an equal classification, both approaches could be assessed.

A third dimension of problems was when they occur. Participants were allowed to ask questions during the conduction of the experiment, for example because they did not know how to model a specific aspect or because something was unclear. These questions could range from minor to major problems. Additionally, errors could be detected in the acceptance test and led to the rejection of a model. Here, minor problems were probably not found. Furthermore, errors could be found in the resulting model. Again, minor problems were probably not detected, as the models were not analysed down to the last detail. Finally, errors might have arisen in the interpretation of the results of the model analysis. Here, errors that led to a permutation of the ranking across clusters could be considered a major error, whereas permutations within a cluster could be considered a minor error. Even though all these occurrences were included in the metric, they should still be differentiated to possibly allow further conclusions.

Using these three dimensions problem area, severity, and occurrence, which are also depicted in figure 3.2, metric M2.1 collected all problems (and errors) and assigned them to the respective

![Figure 3.2: Metric M2.1: Problem dimensions](image-url)
classes mentioned above. Relative values were used, not absolute. A larger group of participants for one approach or a smaller number of finished models are an advantage for the respective approach, as less errors are likely. Thus, the number of modelling errors needed to be averaged over the actual models created.

2.1 Are the approaches comprehensible?

Overview of the metrics To assess how well the participants understood the approaches, several metrics were measured.

First, the number of problems related to comprehension could be derived from metric M2.1. To complement it, the next two metrics measured the understanding quantitatively for both approaches, by measuring the number of acceptance tests needed (metric M2.2) and the number of errors in interpreting the results of the performance predictions (metric M2.3). Metric M2.4 was a qualitative questioning of the participants on the understandability of the approaches.

With the next metrics, details of the Palladio approach should be analysed to directly help assessing the applicability of the approach by evaluating its specific properties. In particular, the participants needed to specify the parameters and their distributions. For both aspects, metric M2.1 provided the quantitative amount of problems. Furthermore, the qualitative evaluation of comprehensibility by the participants was captured in metric M2.5 for the distributions and M2.6 for the parametrisation.

The following enumeration summarises the metrics for question 2.1:

M2.2. Number of times of rejection before acceptance level is reached.
M2.3. Number of interpretation problems
M2.4. Subjective evaluation of comprehensibility by the participants
M2.5. Subjective evaluation of distribution functions by participants
M2.6. Subjective evaluation of parametrisation by participants

Detailed description of the metrics Problems during modelling make problems with the comprehensibility visible. However, not all errors during modelling were related to the comprehension of the approach. It was distinguished in metric M2.1 whether a problem resulted from a lack of understanding of the approach or of the task, or from tools flaws or other reasons. The number of problems related to comprehension were used to answer question 2.1.

Metric M2.2 Number of times of rejection before acceptance level is reached

To complement metric M2.1, the number of acceptance tests before the acceptance level was reached was also measured with metric M2.2. This could give further insight on how hard it was for the participants to correct errors and find additional ones. Again, the metric was averaged over the number of created models. Let $\text{Acc}#_p$ be the number of acceptance tests that
CHAPTER 3. RESEARCH METHOD

participant $p \in P_{a,s}$ needed until his solution was accepted. With this definition, the minimum value for $Acc\#_p$ was 1. Thus, the average number of rejections could be defined as

$$\text{Metric [M2.2]} \quad Rej\#_a = \text{avg}\{Acc\#_p - 1 \mid s \in S, p \in P_{a,s}\}$$

**Metric [M2.3] Number of interpretation problems** The best model was useless if the participants did not understand how to interpret the results the tool generated from it. Thus, the number of errors in interpreting results was measured in metric [M2.3]. An interpretation error occurred if a participant created the model, got the results that allow a correct ranking and ranked the alternatives differently. Each permutation in the ranking was counted as an interpretation error. The metric could be measured like metric [M1.4] but only recognising permutations that were caused by interpretation errors. Thus, I redefined

$$perm'_{u,p}(\text{variant}) = \begin{cases} perm_{u,p}(\text{variant}), & \text{if there is an interpretation error} \\ 0, & \text{else} \end{cases}$$

and built up the rest of the metric equivalently to metric [M1.4] except that I replaced $perm_{u,p}(\text{variant})$ with $perm'_{u,p}(\text{variant})$. I got

$$\text{Metric [M2.3]} \quad prop\text{IntErr}_a = \text{avg}\{propPerm'_{s,u,p} \mid s \in S, p \in P_{a,s}, u \in UP\}$$

**Metric [M2.4] Subjective evaluation of comprehensibility by the participants** Additionally to the quantitative metrics, the participants were directly asked for their assessment of the comprehensibility of the approaches. This helped interpreting the outcome of the above metrics. Thus, metric [M2.4] was a subjective evaluation of comprehensibility by the participants. The actual realisation could be found in the questionnaire in appendix B.4.3. I asked for the comprehensibility of the procedure model (questions 5 and 18) and, for Palladio, the meta model (question 6). Additionally, I asked to grade the comprehensibility of the different concepts of the approaches on a scale from ++ (or 2, i.e. very good) to - - (or -2, i.e. very bad), with the intermediate steps of +, o, - (questions 7 and 19). For Palladio, I also asked whether the division into several roles helps to understand the approach (question 8). Finally, I asked the participants which approach was easier to understand (question 30).

**Metric [M2.5] Subjective evaluation of distribution functions by participants** Two of the main characteristics of the Palladio approach in comparison to other performance prediction approaches are the parametrisation and the usage of distribution functions (see also section 2.2.1). As they are important characteristics of the approach, their influence on the comprehensibility was evaluated separately to find out whether the introduction of these concept complicated the approach, did not affect the comprehensibility of the approach or actually eased the approach.

To assess whether the participants could handle the concept of distribution functions, the number of errors in their specification of the functions was looked at, which were collected in metric [M2.1]. Additionally, qualitative questions were asked on the participants’ opinion on the comprehensibility of the specification of distribution functions in metric [M2.5]. The precise phrasing
of the questions can also be found in the questionnaire in appendix B.4.3. I asked for the understandability of the resulting distributions for the interpretation of the results (question 22) and whether the analysis of the resulting distribution is a better foundation for design decisions (question 23).

**Metric M2.6 Subjective evaluation of parametrisation by participants**  The other main concept of Palladio is the parametrisation of the models. Here, the number of errors in specifying parametrisations can be extracted from metric M2.1. Finally, qualitative questions on the parametrisation were asked to evaluate the participants’ opinion (metric M2.6), which can also be found in in the questionnaire in appendix B.4.3 I asked the participants to evaluate the parametrisation and name advantages and disadvantages (question 9). Additionally, they were asked to estimate the impact of parametrisation for larger and more complex systems (question 10). They were also asked whether the parametrisation eased or hindered the specification of complex branch probabilities, as needed for the bit rate conversion design option of the Media Store system and the initial system of the Web Server (question 12). In these cases, the branch probabilities depended on several parameters and not just one. Additionally, one way of modelling the initial Web Server system involved the calculation of the needed probabilities using the special Bayes’ formula (for details, see [Sac97, p.78]). See section 4.2.2 for more details on these options. Finally, I asked whether potential problems with the parametrisation were due to the concept itself or rather due to the specific concrete presentation in the tool (question 16).

**2.2 Are the tools usable?**  Next to the actual comprehension of the approaches, the usability of the tools might influence the quality of prediction. The following metrics measure the potential problems with the tool.

**Overview of the metrics**  For the usability of the tools, I looked at the number of problems and the actual types of problems measured with metric M2.1. Additionally, the qualitative evaluation of the usability of the tools by the participants is captured.

The following enumeration summarises the new metrics for question 2.2:

- M2.7. Subjective evaluation of the tool usability by participants

**Detailed description of the metrics**  As mentioned for sub-question 2.1, errors in modelling might result from comprehension problems as well as from problems with the usage of the tools. The first area was captured in the previous metrics. Here, the latter influence was measured analogously.

First, the number of questions regarding tool problems as captured during the experiment sessions could be derived from metric M2.1. Typical problems were searched for in the actual recorded questions.
CHAPTER 3. RESEARCH METHOD

Metric [M2.7] Subjective evaluation of the tool usability by participants  Finally, a subjective evaluation of the tool usability by participants collected the subjective problems with the tool in metric [M2.7]. The precise phrasing of the qualitative questions can be found in the questionnaire in appendix [B.4.3]. I asked whether the tools are suitable for a performance prediction and what advantages and disadvantages the participants see (questions 14 and 20). Additionally, I asked for a direct comparison of the tools (question 32). For Palladio, I added a question asking whether it would be helpful to add a textual concrete syntax, e.g. a kind of pseudo code for the SEFFs, for some model parts and if yes, which model parts should be changed (question 15).

2.3 What are further reasons?  There might be other reasons not foreseen when designing this GQM plan but still affecting the quality of the predictions. To find out further issues, the participants were asked for an explanation in almost all questions on the questionnaire in appendix [B.4.3]. In analysing the answers, more reasons might be detected. Additionally, I asked for suggestions how to improve both the approaches and the tools (questions 17 and 21), whether the participants had more trust in the predictions of one approach, if yes, which approach and why (question 29), and which approach the participants preferred and why (question 31).

The resulting metric [M2.8] was a highly vague metric, but it might still lead to valuable results by revealing reasons that otherwise would remain undetected. Thus, it was incorporated in the GQM plan.

M2.8. Analysis of explanations in questionnaire to find additional influences.

3. What is the duration of predicting the performance?

Rationale  Another factor influencing the applicability of a performance prediction approach was the time needed for a prediction. This factor is clearly second to the quality of the prediction: A poor prediction that can be done in a very short time still has no value to the user. However, a good prediction that needs a very long time to be accomplished may prove infeasible in practice and thus be not applicable.

A higher effort was expected for Palladio, as further effort is put into the models to make them reusable and parametrisable, whereas SPE models are created for a single project. Additionally, the specification of more precise distribution functions instead of mean values might lead to additional effort.

Thus, my hypothesis 3 was that the duration for a Palladio prediction is 1.5 times higher as the duration for an SPE prediction. I based this hypothesis on experience from the field of code reuse cost models, where a median relative cost of writing for reuse of 1.5 with a standard deviation of 0.24 over several studies was detected by [Pou96, p.29] in a meta-study. This quantitative hypothesis was statistically tested for using Welch’s t-test [Wel47], which is suitable to compare two distributions that have different variances, and which is available in the R tool [Dal03]. As a significance level, I chose 5%, which is a common value [Sac97, p.198]. However, Welch’s t-test assumes that the samples result from a normal distribution, which was unknown here.
Overview of the metrics  The duration of making a performance prediction included reading the specification \((ra)\), modelling the control flow \((cf)\), adding resource demands \((rd)\), modelling the resource environment \((re)\), modelling the usage profile \((up)\), searching for errors \((err)\) and analysing \((ana)\) for all variants and usage profiles of the system under study. To answer the question for the experiment session, metric \textbf{M3.1} measured the average duration over all participants.

Additional effort for applying the approaches was the training effort, which was measured in metric \textbf{M3.2}. Finally, the participants were asked for a qualitative evaluation of the needed time in metric \textbf{M3.3}.

The following enumeration summarises the metrics for question 3:

- M3.1. Average duration of a prediction
- M3.2. Time needed to solve preparatory exercises
- M3.3. Subjective evaluation by participants on needed time and effort to learn the approaches

Detailed description of the metrics

Metric \textbf{M3.1} Average duration of a prediction  For each participant \(p \in P_{a,s}\), the duration \(d_p\) of making a performance prediction was measured. The duration included the activities mentioned above. The duration was averaged over all participants.

\[
\text{Metric } \textbf{M3.1}: \quad d_a = \text{avg}(\{d_p | p \in P_{a,s}\})
\]

Metric \textbf{M3.2} Time needed to solve preparatory exercises  Next to the time actually needed in the experiment sessions, the time needed to solve the preparatory exercises could be used to compare the time needed for the two techniques (metric \textbf{M3.2}). Whereas time constraints were used in the experiment and might distort the results, the time needed to solve the preparatory exercise was free from this influence. However, the time was measured by the participants and thus could not be verified. Additionally, the time might include time to actually learn the approaches, e.g. by reading documentation.

Let \(E\) be the set of preparatory exercises and let \(pd_{p,e}\) be the duration of participant \(p \in P_{a,s}\) solving preparatory exercise \(e \in E\).

\[
\text{Metric } \textbf{M3.2}: \quad pd_a = \text{avg}(\{pd_{p,e} | p \in P_{a,s}, e \in E\})
\]

Metric \textbf{M3.3} Subjective evaluation by participants on needed time and effort to learn the approaches  The duration to solve the preparatory exercises might not contain the whole effort to learn the approaches. Still, a comparison could be done by asking the participants for their subjective estimation which approach needed more time to learn it. This question was added to the questionnaire in appendix \textbf{B.4.3} (question 24) and formed metric \textbf{M3.3}.
4. What are the reasons for the duration?

Rationale  To further analyse the time needed for a prediction, the factors influencing the time needed were looked at. It was measured how much time was needed for the single activities of the tasks.

It was expected that the modelling was the most laborious part of both approaches. If other parts, e.g. the searching for errors, were more time-consuming, then the applicability of the approaches has to be doubted.

Thus, my hypothesis 4 was that the most time-consuming activity is the modelling.

Overview of the metrics  First, the duration of the single chronological steps of the experiment exercise were measured in metric \textbf{M4.1}. Metric \textbf{M4.2} broke down the overall duration into the duration of the different activities of a performance prediction as introduced above.

Finally, I asked the participants several subjective questions on the reasons for the needed time in metric \textbf{M4.3}.

The following enumeration summarises the metrics for question 4:

- M4.1. Duration of the single steps
- M4.2. Breakdown of the duration to activities
- M4.3. Subjective evaluation by participants on reasons for the needed time

Detailed description of the metrics

\textbf{Metric M4.1} Duration of the single steps  To measure the time needed for the single steps of the tasks, time stamps were introduced. The participants were asked to note when they started and finished parts of the task. A part of the task was doing an activity $act \in \{ra, cf, rd, re, up, err, ana\}$ for a variant of a system $v \in V^s$, $s \in S$. The duration $dact_{v,u,a,act}$ measured the time needed by participant $p \in P_{a,s}$ to do activity $act \in Act$ for variant $v \in V^s$ and usage profile $u \in UP$. The average was calculated over all participants:

\[
\text{Metric M4.1: } dact_{v,u,a,act} = \text{avg}\left(\left\{dact_{v,u,p,act} | s \in S, p \in P_{a,s}\right\}\right)
\]

The questionnaire this metric was collected with can be found in the appendix \textbf{B.3}.

\textbf{Metric M4.2} Breakdown of the duration to activities  To combine how much time was needed in average for the single activities $a \in Act$ that needed to be done over all variants of the systems, metric \textbf{M4.2} averaged the time for each activity over the variants and usage profiles. With this metric, conclusions could be drawn which aspects needs to be improved to improve the needed time:

\[
\text{Metric M4.2: } dact_{a,act} = \text{avg}\left(\left\{dact_{v,u,a,act} | s \in S, v \in V^s\right\}\right)
\]
Metric M4.3: Subjective evaluation by participants on reasons for the needed time

Finally, I added questions on the reasons for the needed time to the qualitative questionnaire (cf. appendix B.4.3). First, I asked which approach was more time-consuming to apply and why (question 25). To assess the influence of the used tool, I asked which tool was faster to use (question 27).

For Palladio, I asked whether the parametrisation eased the specification of the SEFFs or whether it was an additional effort (question 11). Additionally, I asked how the participants estimated the effort if the Palladio roles were actually assigned to several people (question 26).

Furthermore, I asked the participants to assess the automated transformations available in Palladio, as used the broker lookup alternative (cf. section 4.2.2). Here, the effect on the needed time was an issue, however, the participants were asked a more general question to allow further insight in their opinion.
4 Design and Conduction of the Experiment

The study was conducted as a controlled experiment, to investigate the applicability of Palladio and SPE with participants who are not the developers of the approaches. 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. Here, the independent variable was the approach used to make the predictions. Observed dependent variables were the created models, assessed using several metrics, and the duration of making a prediction.

This chapter describes the experimental set-up, as well as the required preparations. In section 4.1 I describe the participants and their training. In section 4.2 I describe the actual experiment. Finally, I discuss the validity of the experiment in section 4.3.

4.1 Participants

When designing an experiment, the participants are the first to consider. In this experiment, the students of the course ”Ingenieurmäßige Software-Entwicklung” (Engineering Software Development) at the University of Karlsruhe in summer term 2007 were asked to take part. Thus, the participants of the experiment were students of 3rd and 4th year. All were male computer science students, with the result that I can use the male third person pronoun for the participants in this thesis without discriminating against any women.

To further assess the participant’s abilities for this experiment, a questionnaire was issued. The questionnaire asked for programming experience as well as software design experience as an indicator for software engineering skills. Lutz Prechelt even stated that his experiments showed that experience (except for a certain training with the techniques) has no correlation with the performance in the experiment, and that the individual mental abilities (for example measured by SAT scores) have a far better correlation. [Lutz Prechelt, during a review session for this experiment design, 02/22/07, Schloss Dagstuhl]. As a SAT score or an equivalent measure were not available, I asked for a self assessment of programming skills, which was an acceptable alternative according to Prechelt (ibid.).

The results of the questionnaire, characterizing the participants, can be seen in the figures 4.1(a) to 4.2(a). In figure 4.1(a) the self-assessment of the programming skill is shown. The participants were asked to classify themselves belonging to the top 5% (95-100), the next 15% (80-95), the next 30% (50-80) and the lower half (0-50) based on their programming skill. The particular scale was suggested by Prechelt (ibid.). As shown in the figure, no participants classified
themselves as being in the top group. Most participants assessed themselves as being in the lower upper half. All participants had completed at least 5 semesters of study, thus they were advanced students. The number of semesters is also depicted in figure 4.1(b).

Most participants had several years of programming experience (cf. figure 4.2(a)). 4 participants stated that they had little programming practice, 8 that they had intermediate programming practice and 6 that they had much programming practise. Most participants had already designed several systems with less than 5000 lines of code, 7 of them named 10 or more, and half of the participants designed one or more systems with more than 5000 lines of code. Furthermore, less than half of the participants stated that they had some experience in performance analysis (cf. figure 4.2(b)). For this assessment, the questionnaire suggested that "little" means having analysed small systems of less than 1000 lines of code, that "intermediate" means having analysed medium-sized systems of less than 5000 lines of code, and "much" means having analysed larger systems with 5000 lines of code or more, or even a job in this field.

Furthermore, all except three participants took the software engineering course, thus being familiar with UML notations. Only one participant did not visit a software engineering related lecture before, but he stated to have much programming experience. The others visited other software engineering related courses and probably were familiar with the notation. Figure 4.3 shows how many participants visited the software engineering related courses at the University of Karlsruhe.

Most participants had no or little experience with performance analysis (cf. figure 4.2(a)). One participant stated to have medium experience with performance analysis, one stated to have much experience with performance analysis, but profiling and tuning only. Thus, a preparation of all participants was required. This also led to a similar standard of knowledge of all participants, which is advantageous for the significance of the results [Pre01 p.112].
Figure 4.2: Programming experience and experience with performance analysis of the participants

(a) Programming experience

(b) Experience with performance analysis

Figure 4.3: Visited courses
CHAPTER 4. DESIGN AND CONDUCTION OF THE EXPERIMENT

All things considered, the participants all had a base competence. In most cases, their competence could be compared to a competence of a professional whose main interest is not in performance analysis.

A common objection to experiments in software engineering involving student participants is that the results cannot be transferred to ”real” software development. According to [Pre01], this objection may be true, but in most cases is exaggerated. He argues that (1) the difference between advanced students and professionals is not very great and that (2) this difference is not relevant, because not the absolute achievements of the participants is measured in an experiment, but the change of the achievement when changing the experimental variables. However, this is only applicable if the working method of less competent participant is not different to that of a competent participant. To ensure the working methods do not differ, the task must not be to complex and the participants must not be too inexperienced with software engineering in general or the specific kind of exercise, as the task would otherwise ask too much of them.

In addition, experience in software development is often over-estimated: In this study, the experience within a specific application domain was irrelevant. Besides, students have a similar individual background. Hence, outliers due to individual performance are less likely.

Considering all afore-mentioned aspects, the results of this experiment are transferable to the same situation involving professional software engineers, if not professional performance analysts.

However, in [Pre01], Prechelt states that the competence is not the most important factor for the qualification of participants for an experiment. The more important aspect is that the way of solving of the experimental task is realistic and the same for all participants and in reality. Therefore, participants must be trained with the techniques under study and have basic software engineering skills. Very inexperienced students who have no software engineering knowledge will likely have a different way of solving tasks than more experienced students, this is why beginners are not suited for an experiment. Several experiments, in which well-trained students performed better than both professionals and less qualified students, show that this aspect is even more important than competence [Pre01, p.95]. The way of solving the task can also change if the task is to complex, too particular, or too time-consuming [Pre01, p.111].

The motivation or missing motivation of the participants can be a further problem for an experiment. Here, the strongest threat is differences in the motivation of different groups [Pre01, p.115]. It is dangerous if the participants know what is tested, because they might be more motivated if they apply the new technology or a technology the experimentators are biased towards. Here, the experimentators have to ensure that they do not present the technologies in a biased way.

To ensure that the participants took their tasks seriously, the participation in the preparation was compulsory. For each preparatory exercise, the participants needed to achieve 60% of the potential points, and were only allowed to fail a single exercise.

For the experiment, the award was similar. However, the participants should not be under too much pressure in the experiment. Thus, only a base quality of the models was graded and not the final results, with interpretation and conclusions. If the final models were of a sufficient quality, full marks were awarded. Additionally, models that were not finished were not graded. In doing so, the time needed and potential tool flaws did not affect the grading. Overall, the
achieved points in both preparatory exercises and experiment made up 2/3 of the course grade, if a grading was requested by the student.

Additionally, the results in the preparatory exercises can be used to assess the competence of the participants [Lutz Prechelt, during a review session for this experiment design, 02/22/07, Schloss Dagstuhl], if the motivation to solve the preparatory exercises is similar to the motivation in the experiment.

4.1.1 Preparation

The participants had to train the approach beforehand, as untrained participants would have to use a main part of the time in the experiment session to learn the approaches, thus leaving no time to actually work on the task. Additionally, problems with understanding the approaches can be discussed beforehand. To ensure that the participants are familiar with the approaches, training session were established.

The participants in the experiment were trained in applying SPE and Palladio during the 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 lecture introduced the course and its organisation. The second lecture was dedicated to foundations of performance prediction and CBSE, the third introduced the two tools. 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 metamodel which allows reusable prediction models. Note, that this also shows that reusable models require more training effort. 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 the SPE approach and five the Palladio approach. The exercises can be found in appendix A.5.

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

Competence tests during the preparation phase ensured a certain level of familiarity with the tools and concepts. Firstly, the results of preparatory exercises were examined. Additionally, four short test were conducted in the lectures. Participants who failed two preparatory exercises or a short test could not take part in the experiment.

The teaching process ended with a questions and answers session where the students could ask final questions.

The preparation did not only train the participants, but tested their abilities as well as the formulation of the task and thus fulfilled the role of a pretest [Pre01]. With a pretest, the learning effects during the experiment is minimized, as the participants learn during the pretest. Learning effects during the experiment itself may invalidate the results of the experiment. Additionally, the pretest could be used to assess the participant’s abilities and balance the two groups (each
applying one approach) so that the ability of the groups were about the same. In this experiment, the two experiment groups were set up based on the participant’s results in the pretest, so that each group had stronger and weaker participants.

4.1.2 Preparatory Exercises

The preparatory exercises can be found in the appendix. The first exercise trained basic concepts of component-based software engineering. Exercise 2 trained the fundamental usage of the tools for both approaches (2a SPE, 2b Palladio). The participants had to install the tools and create a simple project following detailed instructions. After that, the exercises 3, 4, 5, and 8 trained SPE and the exercises 5, 6, 7, and 8 trained Palladio.

Exercise 3 trained a simple performance analysis for SPE. Use cases and sequence diagrams were given and the execution graph had to be extracted from this information. The overhead matrix was given in the task description, as well as performance annotations for the single steps of the control flow. The participants were asked to analyse the system for a single user scenario. Exercise 4 trained a more complex example. Here, the participants needed to calculate some performance annotations from given information and create the overhead matrix themselves based on given information. Additionally, the system was a distributed system. A multiuser analysis, using both the analytical approach and simulation, was required from the participants. Exercise 5b included the correction of exercise 4 and an analysis how much more users can use the system without increasing the response time by more than 10% and how much users the system can handle overall.

Additionally, exercise 5a trained the creation of a Palladio component repository with the same example system as given in exercise 3 and some additional information on interfaces and signatures. In exercise 6, the participants were asked to add RDSEFFs to the components of exercise 5. The control flow was given as a sequence diagram, additionally performance annotations and information on the resource environment were given. Finally, the participants were asked to simulate the system and interpret the resulting histogram.

Exercise 7 trained the specification of parameters, stochastic expressions, and distributions in Palladio. Participants again used their solution for exercise 6. Again, the participants were asked to simulate the system and describe the differences towards the analysis of exercise 6.

Exercise 8 included again both an SPE (8a) and a Palladio (8b) task. A component-based groupware system was to be analysed with both approaches, thus revising the concepts of both approaches. Two design options were analysed by the participants, who then had to choose the best option in terms of lowest response time for the given usage profile. For SPE, global parameters were introduced in this exercise, as were local parameters for Palladio.

4.1.3 Results of the Preparation

Two students were excluded from the experiment because they failed a short test or two preparatory exercises. The other 19 participants achieved between 133 and 161 out of 171 points in the preparatory exercises. The distribution is shown in figure 4.4.
I balanced the grouping of the participants based on the results in the preparatory exercises: I 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. I chose not to use a counterbalanced experiment design, because in that case, I would need to further divide the groups, which would hinder the balancing between the groups. I expected a higher threat to validity from the individual participant’s performance than from sequencing effects.

4.2 The Experiment

This section describes the experiment set up. First, I describe the experiment plan. After that, the experiment tasks are described in section 4.2.2. The actual execution of the experiment as well as occurred problems are described in section 4.2.3.

4.2.1 Experiment Plan

The experiment was designed as a changeover trial as depicted in figure 4.5. 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 participant worked on two tasks in the course of the experiment (inter-subject design) and used both approaches. This allowed to collect more data points and further balanced potential differences in individual factors like skill and motivation between the two experiment groups. Additionally, using two tasks lowered the influence of the concrete task and increased both the internal validity [Pre01, p.124] as well as the generalisability, which is most threatened by specific characteristics of the single experimental tasks [Pre01, p.154].
Before handing in, the participants’ solutions were checked for minimum quality by comparing the created models to the respective reference model. This acceptance test included the comparison of the predicted response time with the predicted response time of the reference model as well as a check for the well-formedness of the models. An acceptance test has two advantages. Firstly, it ensures that all handed-in solutions have a minimum quality and in doing so, allows to draw conclusions on the time needed to make a prediction [Pre01, p.138]. Without acceptance test, potentially resulting incomplete solutions cannot be used for the interpretation of the results. Secondly, if the participants know that their solutions will be checked before accepted, they might put more effort in the solution [Pre01, p.138], as they cannot hand in any solution.

The participants only had a limited time for completing the tasks. I conducted two sessions, each with a maximum time constraint of 4.5 hours. There were several reasons for this decision. Firstly, it is an organisational constraint. The participants should be under surveillance during the whole task, to avoid the exchange of knowledge between the participants. Thus, the participants cannot stay in the chosen room as long as they want. Still, a longer time limit was organisationally possible. However, I wanted to avoid effects of tiredness and resulting “slips of the pen”. Finally, time constraints are overall useful because they represent the time pressure always existent in industrial settings [Pre01, p.139].

However, the combination of acceptance tests and time restrictions can be problematic [Pre01, p.139]. There might be participants who are not able to produce an acceptable solution in time. This results in less data points and hinders the interpretation of the remaining ones.
Additionally, if all participants use the entire time, the duration cannot readily be used for interpretations.

Because of these problems, the time limitations are not fixed, but can (and were) relaxed during the experimental task if the need arises and the majority of participants cannot finish in time.

To also collect qualitative and subjective data on experiment, qualitative questionnaires were issued after each session and a week after the last session. The questionnaires after each session targeted the experiment task and asked for problems in it, e.g. whether the time limit was too small. See appendices B.4.1 and B.4.2 for the two questionnaires. The last questionnaire asked questions on the comprehensibility of the concepts, on the tools and especially questions comparing the two approaches (cf. appendix B.4.3). Next to qualitative data, these post-mortem questionnaires also help to assess the influence of problems in the experimental task on the results [Pre01, p.140].

4.2.2 Experimental Tasks

To be applicable for both SPE and Palladio, the experiment tasks could only contain aspects that can be realised with both approaches (cf. section 2.3.2). For example, the tasks could not make use of the separate roles of Palladio and performance goals related to the actual distribution of the response time (“90% of the time, the system should answer within 2 seconds”), which is available in Palladio only, were not evaluated.

Both experiment tasks had similar set-ups. The task descriptions contained UML-like component and sequence diagrams documenting the static and dynamic architecture of a component-based system. The sequence diagrams additionally contained performance annotations. The resource environment with servers and their performance properties was documented textually. The systems in both tasks were prototypical systems that had been specifically designed for this experiment. For each system, two usage profiles were given, to reflect both a single-user scenario (UP1) and a multiuser scenario leading to contention effects (UP2). Additionally, they differed in other performance relevant parameters. With the two usage profiles, different requirements are reflected, that may already be known during the design phase. Different usage within a time period (day, week, ...) can be reflected or anticipated change of usage, e.g. an increase of the number of users.

In addition to the initial system, five design alternatives were evaluated. This reflects a common task in software engineering. Four of them were designed to improve the performance of the system, 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 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 participants were asked to first read the task description and then rank the design options without any further analysis. In doing so, I wanted to find out whether the design options could be correctly assessed without any approach being applied, which can be seen as a very limited control group. If the participants had been able to manually estimate the correct design decision,
the task might have been too simple and easy to see through. The results of the initial ranking are discussed in section 5.2.3 with the construct validity.

After the initial ranking, the participants needed to model the initial system and analyse its response time for the two given usage profiles. Before proceeding to the design options, they were asked to check with the experimentators whether the solution is acceptable. After modelling and analysing each design option, they had to again pass the acceptance test. Finally, the participants were asked to draw conclusions. First, they assessed which design options were advantageous for the response time or, in the case of the fifth design option, i.e. the broker alternative, whether it only increased the response time by 10%. Next, they created a ranking for the design options based on their usefulness. Note that participants might have chosen a ranking that not only based on the smallest response time. However, they were asked to give the reasons for their decision, so this was visible from the reasons.

The feasibility of the tasks was checked by presenting it to four graduate students who also took part in the preparation but were not further connected to the experiment itself.

Media Store

In the first task, the participants were asked to analyse a web-based system called Media Store. With this system, users bought and stored their mp3 files over the Internet. The system supported two use cases: To upload single mp3 files to one’s storage and to download up to 12 files.

Figure 4.6 shows the components initially used in this system, their assembly and the sequence diagram of the download use case. More details and figures on the task can be seen in appendix B.1.1.

The Media Store was chosen because it was a typical, even if simple, multimedia web application, which is often found nowadays. It consisted of a user interface (here, the WebGUI
components responsibility), business logic (encapsulated in the MediaStore and Digital-Watermarking components) and a database (the AudioDB component).

Two usage profiles were given, one for a single user scenario, and one for the multiple access of several users, resulting in contention effects. The usage profiles contained information on the frequency how much the two use cases were used, the number of mp3 files to be downloaded at once, the size of the mp3 files used, and the encoding of the mp3 files (which is important for a later design option). For Palladio, all information except the frequency of the use cases were given as distributions, for SPE a mean value was given. In the second usage profile (multiuser), the number of files and the frequency of the use cases was changed.

All components were allocated on a single server. The performance relevant data for CPU and hard disk drive were given in the task description.

The systems performed calculations such as the parsing of the HTTP request and, for downloaded files, a digital watermarking of the files. Additionally, the hard disk drive was used when reading and writing files from and to the AudioDB component. For both usage profiles, the performance critical part of this system was the hard disk drive access. Thus, the number of mp3 files downloaded and the their file size heavily influenced the performance.

The Media Store task came with 5 design alternatives presented below, all of them potentially improving the system, however, also coming with drawbacks. The detailed description of the design options with UML diagrams showing the static and dynamic changes, can be found in the experiment task in appendix [B.1.1]. Here, I present each alternative by first describing it, then giving its influence on the performance (derived from the reference models as presented later in this section, and of course unknown to the participants) and finally arguing why the alternative is suitable in this task. The assessment of the influences on the performance is always based on the two usage profiles and the resource environment presented above.

**Introduction of a cache component (v_{1MS}^C):** For this option, a new Cache component was introduced and chained between the MediaStore and the AudioDB component. The cache kept a certain amount of mp3 files in memory and thus reduced the number of hard disk drive accesses. However, the check of the availability of a file in the cache needed some calculation done by the CPU. The cache hit ratio was given in the task description.

As the hard disk drive access was the bottleneck of the system, this option greatly improved the performance of the system, as it replaced the long hard disk drive access with a rather short calculation for checking the availability in the cache.

As the introduction of a cache was a very common action to improve the performance of a system, this alternative reflects real decisions very well. A slight drawback was that the use of a cache component in this setting was only connected to a very small trade-off, which made the outcome not very surprising.

**Use of a database connection pool (v_{2MS}^{DB}):** For this option, the AudioDB component was replaced with the PoolingAudioDB component, that used a database connection pool to access the internal database. Thus, the accesses to the database needed less computations and the number of concurrent accesses to the hard disk drive was reduced, thereby
reducing contention effects. However, only a certain amount of transactions could be executed concurrently, other users had to wait.

The impact on the performance, however, was only slight. The hard disk drive as the bottleneck of the system was not relieved of files to read and write. The reduction of contention effects and the lower computational effort only led to a slight improvement of the response time.

Still, this alternative was a realistic and useful one, because the use of database connection pools is widespread to improve performance. However, this alternative showed that the characteristics of the system and its usage can lead to a widespread technique not being very useful in a special case. Additionally, passive resources could be analysed with this design option.

**Use of a second server** ($v^M_S$): For this option, a second server was added and the AudioDB component was allocated on it. The second server provided both computing power and a second hard disk drive. Additionally, a network link between the two servers was introduced, which was a drawback having an additional latency.

The use of a second server worsened the performance for the single user case, as one could expect. The network caused an additional delay, and one server was always idle while the other one was computing. For the multiuser case, the improvement was only very slight. As all accesses to the hard disk drive were now executed on the second server, the first hard disk drive became obsolete. Only the needed computations were shared between the servers.

The introduction of more hardware is also a very common technique to improve performance. This alternative showed on a admittedly very simple way that the pure adding of computational power does not always help. Additionally, this option allowed to analyse distributed systems and network communication.

**Re-encoding and reduction of bit rate** ($v^M_4$): For this option, a new Encoding component was introduced and the MediaStore component was replaced by a Encoding-MediaStore component calling the Encoding component before sending files to the AudioDB. The Encoding component reduced the bit rate of mp3 files with high bit rates (e.g. CD quality) by re-encoding them and thus reduced the file size of files in the database. The drawback was a computational effort for the encoding.

Of course, the usefulness of this option strongly depended on how much the files could be compressed and how many large files were uploaded. For the values assumed here, i.e., reducing the file size by averaged 17%, the reduction of the bit rate was very useful, because it traded in this setting expensive hard disk drive accesses against in this setting relatively cheap computations.

Again, the so-achieved compression is a common technique to improve performance. It was interesting to analyse because the effects relied heavily on the assumed parameters and the actual bottleneck resource of the system. Additionally, this option added more complexity to the control flow than the other alternatives did, and thus might lead to different results for the use of parametrisation of Palladio.
**Broker lookup (v₃⁵₈):** This last design option was not introduced to improve the performance, but to add more maintainability and dynamism to the system. The alternative was to add a middleware broker each component gets its communication partner from to the system. Assuming that the initial system had used dependency injection, this added more flexibility to the wiring of the components and allows dynamic changes of the assembly. However, the broker look up was more costly than a dependency injection call of a component.

The requirement here was that the response time of the system was not increased by more than 10% by the introduction of a broker lookup. This requirement was slightly not fulfilled for the single user case, but easily fulfilled for the multiuser case, because the contention effects on the hard disk drive increased the response time greatly.

The use of this alternative was to analyse Palladio’s built-in model transformations. Such an aspect of the configuration of the middleware is a typical example where such transformations are used.

**Reference Model** I modelled the Media Store using SPE and Palladio, to be used as a reference for the acceptance test and for the comparison with the models created by the participants.

After modelling, I analysed the different combinations of design options and usage profiles, resulting in 12 results. For Palladio, I set a simulation time of 5000 simulated seconds in the PCM Bench, as this delivered fairly stable results. For SPE, I used the analytic solution of the models, as the results of simulations were very scattered and varied a lot. The highest measurement was easily twice the lowest measurement. Additionally, the simulation did not support distributed systems either (cf. section 2.3.1).

In the following, results of the analyses are presented. At the same time, I identify the design option ranking using the predicted response times of the reference model. For SPE, I used \( \text{refMeanResp}_{v,u,SPE} \). For Palladio, I looked at the cumulated response time distribution and chose the one with the largest integral as the best one. If the two graphs are very similar, this decision could not be made, because the graphical representation did not include values for the intervals. In that cases, I further used \( \text{refMeanResp}_{v,u,Pal} \) to assess the better design option.

For the Media Store system and usage profile 1, the best two design options had very similar predicted response times for both approaches, the mean values only differing in some milliseconds. The better design option was the change of the bit rate (v₃⁵₈). In the SPE-ED predictions, this option was deemed 12 ms faster that the second-best option, the cache (v₃¹₈, see figure 4.7(a)). In the Palladio prediction, the difference was smaller. Looking at the cumulated density function (CDF), the best alternative cannot readily be distinguished (see figure 4.8). The mean of the distribution for the bit rate option (v₃⁵₈) was only 1.4 ms lower than the mean for the cache option (v₁¹₈).

For usage profile 2, the best two design options were likewise very similar in respect to predicted response times. Here, the cache option (v₁¹₈) had a little faster response time: In the SPE-ED predictions, the difference to the bit rate conversion variant (v₃⁵₈) was 11 ms (see figure 4.7(b)). In the Palladio prediction, the difference was again smaller. Again, looking at the CDF, the best
Figure 4.7: Predicted mean response time of the reference for the Media Store system using SPE-ED

Figure 4.8: Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 1 using Palladio
CHAPTER 4. DESIGN AND CONDUCTION OF THE EXPERIMENT

Figure 4.9: Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 2 using Palladio alternative cannot be read off (see figure 4.9). The mean of the distributions was lower for the cache option \( v^{MS}_1 \) by 2.7 ms.

Web Server

In the second session, the participants were asked to analyse a Web Server system with limited functionality. A single use case was supported: Users request an HTML page, possibly containing one or more multimedia objects. To retrieve multimedia objects, separate subsequent requests were issued to the Web Server (cf. figure 4.10(b)). The participants had to determine the response time for the whole use case, not just a single request.

Figure 4.10(a) shows the components initially used in this system and their assembly. More details and figures on the task, e.g. the detailed sequence diagrams of handling of the request, can be seen in appendix B.1.2.

The Web Server was chosen because it represents the general application in the web in a simplified form. It comprised business logic (in the generation of dynamic content) and database accesses. Additionally, it featured a slightly more challenging control flow, as the Web Server components needed to analyse the request and find the responsible component to answer it, depending on whether static or dynamic content was requested.

Again, two usage profiles were given, one being a single user scenario and one a multiuser scenario, resulting in contention effects. The usage profiles contained information on the frequency of static and dynamic HTML pages, on the number of multimedia objects requested per page, and on the frequency of static and dynamic multimedia objects. Additionally, the resulting file
sized were given for dynamic content (HTML and multimedia). For the multiuser usage profile, the frequencies of static of dynamic content and the number of multimedia objects were changed.

All components were allocated on a single server and again the performance relevant data for CPU and hard disk drive were given in the task description.

The system performed computations such as parsing the request, finding the responsible handling component and possibly generating dynamic content. Additionally, the hard disk drive was accessed to retrieve static content. For both usage profiles, the CPU was the bottleneck resource. In the single user case, the hard disk drive was idle in average 96% of the time, whereas the CPU was used in average 96% of the time. In the multiuser case, up to 16 users used the CPU concurrently, whereas only up to 3 users used the hard disk drive concurrently. Thus, the amount of dynamic, computational intensive content is a critical performance parameter for this Web Server.

As the Media Store, the Web Server came with 5 design alternatives, all again having tradeoff characteristics. The detailed description of the design options with UML diagrams showing the static and dynamic changes, can be found in the experiment task in appendix B.1.2. Here, I present each alternative by first describing it, then giving its influence on the performance (derived from the reference models as presented later in this section, and of course unknown to the participants) and finally arguing why the alternative was suited for this task. The assessment of the influences on the performance was always based on the two usage profiles and the resource environment presented above.
**Cache for dynamic content (v_{1}^{WS})**: For this design option, the `ContentGenerator` component was replaced by an `AcceleratedContentGenerator` component, that included an internal cache for the created dynamic content. The cache kept a certain amount of dynamically created content in memory. If the same request came in twice, it was answered by the cache and no new content needed to be generated. As dynamic content was susceptible to changes, the hit rate of the cache was rather low. The check of the cache needed CPU computations, on the other hand CPU computation for generating dynamic content was saved if there was a cache hit.

As the effort for checking the cache for all requests was still lower than the otherwise needed computational effort for the cached dynamic content, the introduction of cache greatly improved the performance.

The design option was useful, for the same reasons as the Media Store cache option: It is very common, even if here, it did not have a surprising outcome.

**Broker lookup (v_{2}^{WS})**: The broker lookup alternative was the same for the Web Server as for the Media Store (see page 65).

Here, the use of the broker increased the response time clearly by more than 10 % for both usage profiles, and thus should not be used with the given requirements.

**Paralleled logging (v_{3}^{WS})**: In the initial system, the logging was sequentially included in the control flow. However, the handling of the requests did not have to wait for the hard disk drive writing the log information (it was assumed that there is no caching by the operating system or the like). In this alternative, the logging was done in parallel to the rest of the control flow. The drawback was an additional computational effort to create a new execution thread for logging.

This alternative led only to slight decrease in response time for the single user case. The cost for logging were not very high and only hardly suffered from contention effects.

This alternative was useful because it used at the parallelism capabilities of the approaches and allowed to analyse them. Here, parallelism between two different resources were looked at, one does not have to wait for the other. The idea of paralleling the control flow might become even more important in the future computing, especially when it comes to multi core machines [ABC+06], i.e. parallelism within a single resource. However, such analyses are neither possible with SPE nor with Palladio at this point of time.

**Use of a second server (v_{4}^{WS})**: For this option, a second server was added and the DynamicFileProvider and the ContentGenerator component were allocated on it. In contrast to the Media Store system, here the second server provided a twice as fast CPU and a second hard disk drive. As with the Media Store, a network link between the two servers was introduced, which was a drawback having an additional latency.

Because of the more powerful CPU, this alternative greatly improves the performance, even for the single user scenario. For the multiuser scenario, additionally, the utilisation of the CPU and therefore the contention effects decreased.
Again, as for the Media Store, the change of the resource environment only is a common technique and thus useful to look at here. Additionally, it represents a distributed system and uses the network.

**Use of a thread pool ($v_{WS}^W$):** In this design alternative, the Dispatcher component was replaced by a PoolingDispatcher that used a thread pool to assign threads to incoming requests. The initial system was assumed not to have any thread pool capabilities, but to create new threads for every incoming request. Generally, the use of a thread pool decreases the contention within the system, however, the users requests have to wait if all threads are already occupied. Additionally, the overhead to create new threads is saved.

For this Web Server, the use of a thread pool was not very advantageous. For the single user scenario, the response time improved slightly, because the overhead for creating a new thread was saved. For the multiuser scenario, the effect was a little larger, as the contention was reduced, especially the predicted maximum response time was lower. However, overall the effect is small.

The use of this design option is again reflecting a common measure to improve performance, even if in a very simplified way. Usually, the question for such systems is not whether to use a thread pool, but rather to choose the optimal pool size [CGLL02]. Additionally, this option included the use of passive resources in the Web Server task.

**Reference Model**  As with the Media Store, I modelled the Web Server using SPE and Palladio, to be used as a reference for the acceptance test and for the comparison with the models created by the participants.

Before describing the results of simulations and analyses, I first explain two ways how to model the system with SPE, as they came with different problems. As the usage scenario can include more than one call to the system, there are several possibilities how to model this in SPE.

The first option was to only model the internal behaviour of the web server (as shown in figure 4.11(a)) and consider the multiple requests per user by increasing the arrival rate of users. To give an example: If one user per second accesses the system and requests an HTML page with two multimedia objects, one can model the request handling only and multiply the arrival rate by 3 (1 HTML + 2 multimedia). However, there were consequences to this approach: In the control flow, it was first the distinction between dynamic and static requests to determine the responsible component. After that, the responsible component delivered either an HTML page or a multimedia object. As there were separate probabilities in the usage profiles for an HTML page being dynamic and a multimedia page being dynamic, the probabilities for an arbitrary content to be dynamic had first to be calculated before being used in the model. As the probabilities were dependent on each other, Bayes’ Rule for conditional probabilities had to be applied (for details on Bayes’ rule, see [Sac97, p.78]). As the application of the rule might not be apparent, I include the needed calculations in appendix B.1.2, page CCIX.

However, there was an alternative way of modelling. The usage profile could be unreeled to form a longer execution graph (as shown in figure 4.11(b)). Here, first the actions when an HTML page was requested were modelled, and then, in a loop, the same actions were repeated, only with different performance values. If one knew how to copy and paste execution graphs,
(a) Only model the internal HTTP handling (Bayes’ Rule)  
(b) Rolled out the usage profile

Figure 4.11: Two ways of modelling the Web Server usage profile
this way allowed easier modelling. However, the drawback was that the execution graph did not only represent a model of the software (as intended) but included the behaviour of the user. It is apparent that any change in the usage scenario would be hard to implement in the models.

After modelling, I analysed the different combinations of design options and usage profiles, resulting in 12 results. For Palladio, I set a simulation time of 5000 in the PCM Bench. For SPE, I used the analytic solution of the models.

In the following, results of the analyses are presented and the design option ranking is identified. For the Web Server system, the difference of best and second-best design option was more pronounced for both approaches and usage models. Figure 4.12(a) shows the predicted response times for the reference model for usage profile 1 using SPE-ED. The alternative featuring a second server ($v_{WS}^{4}$) was clearly the one with the lowest response time. The same result can be seen in figure 4.13 for the Palladio approach. The graph of the cumulative density function (CDF) of the response time for the second server variant ($v_{WS}^{4}$) mostly is above the CDF of the cache variant ($v_{WS}^{1}$).

For usage profile 2, the differences between best and second-best alternative was even more pronounced for both approaches. Again, the variant introducing a second server ($v_{WS}^{4}$) had the best predicted response time, the cache option ($v_{WS}^{1}$) was second-best. For the SPE approach, the predicted response time for the second server option ($v_{WS}^{4}$) was 16.6 ms lower (see figure 4.12(b)). For the Palladio option, the CDF of the response time for the second server variant ($v_{WS}^{4}$) always was above the CDF of the cache variant ($v_{WS}^{1}$), see figure 4.14.

### 4.2.3 Execution of the Experiment

First, I introduced the participants to the experiment and explained the regulations. The participants each received four sheets: One contained the experiment task (see appendix B.1.1 for the Media Store and appendix B.1.2 for the Web Server). The participants were first asked to read the task description and give a first estimation on the ranking of the design options (cf. appendix B.2.1 (Media Store) and B.2.2 (Web Server)). To document the duration of the activities given in metrics M4.1 and M4.2, they additionally received a sheet to note the times
CHAPTER 4. DESIGN AND CONDUCTION OF THE EXPERIMENT

Figure 4.13: Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 1 using Palladio

Figure 4.14: Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 2 using Palladio
in a predefined way (cf. appendix B.3). After the experiment, they received the respective qualitative questionnaire (cf. appendix B.4.1 (Media Store) and B.4.2 (Web Server)).

Drinks and food were provided for free. The sessions took place in a university computer lab. Four members of our chair were present to help with problems with the tools, the exercise, and the methods, as well as to check the solutions in the acceptance tests. This might have distorted the results, because they might have influenced the duration. The more problems were solved by the experimentators, the less time the participants might have spent on solving them themselves. To avoid this effect, the participants were asked to first try to solve problems on their own before consulting the experimentators. To be able to assess a possible influence of this help, I documented all questions and answers as well as all rejections in the acceptance tests (see appendix C.3 for the corresponding sheets).

Problems

Because many participants did not finish the task of the first session within 4.5 hours, the time restriction was loosened afterwards and they were allowed to work another 2.5 hours. During the first session, it became clear that three participants using Palladio were not properly prepared, as they needed a lot of basic help or were not able to finish even the initial system prediction. Thus, the results of these three participants cannot be used. All other participants modelled the initial system and at least one design alternative. Overall, three of the remaining seven participants using the Palladio approach were able to finish all design alternatives, whereas seven of the nine participants using SPE did so.

In the second session, the time restriction was loosened, too, and the participants were allowed to work another 2 hours. One participant did not attend the second session due to personal reasons, thus, only 18 students took part. Again, two of the three participants mentioned above, now using SPE, were not well prepared enough to properly solve the tasks. Their results are not included in the analysis of the metrics. Because these two participants failed using both approaches, omitting their results does not advantage one of the approaches. The other eight participants using SPE finished within the extended time, as well as six of the eight participants using Palladio.

SPE-ED Problems One participant deliberately reduced the network speed. He argued that a certain amount of the network traffic is caused by protocol data and is not available for usage data. However, as the predictions were not compared to a measurement of real implementation, but to the reference model, this improvement led to an error, as it was not contained in the reference model. Thus, the network speed was set back to normal before using the results.

Palladio Problems The participants used different simulation times, which affects the results, especially for the usage profile 2 with contention. Thus, the maximal values vary.

As the resulting data of a Palladio simulation was stored in very large database files, the participants were not asked to hand them in. Thus, the simulations needed to be rerun for the
participants models. This allowed to use a fixed simulation time for all predictions and eliminated errors due to different simulation times.

One participant included the cache component for design option 1 in the Media Store system using the composite diagram, but because of a bug in the synchronisation between diagram and model, the inclusion was not correctly updated in the model. The resulting high response time was detected in an acceptance test, but the reason could not be found at that time. As the participant had actually intended to include the cache, and as it was detected in the acceptance test, the problem was afterwards handled and the corrected predictions are used.

Another participant mixed up DoublePMF and DoublePDF in his models. An experimenter pointed this mistake out in the acceptance test, which is documented. The next acceptance test was passed. However, the final models still contain a DoublePMF instead of a DoublePDF for the Web Server file sizes. Either the participant did not correct the model, which was not detected in the second acceptance test, or a wrong version of the model was finally saved. Again, as it was detected in the acceptance test, the models were afterwards corrected and the corrected predictions are used.

4.3 Validity of this Experiment

The results of an experiment are only useful, if they are valid. Thus, the experiments validity should already be included in the planning of an experiment [WRH+00]. The validity of an experiment is usually classified into four types, namely conclusion, internal, construct and external validity, which was firstly introduced by [CC79]. These concepts can be mapped to the different steps involved when conducting an experiment, as taken from [WRH+00] and the numbers used in the following definition are annotated in figure 4.15 which is a revision of figure 3.1 already presented in section 3.1.1.

**Conclusion validity** "This validity is concerned with the relationship between the treatment and the outcome. We want to make sure that there is a statistical relationship, i.e., with a given significance" [WRH+00, p.64].

**Internal validity** "If a relationship is observed between the treatment and the outcome, we must make sure that it is a causal relationship, and that it is not a result of a factor of which we have no control or have not measured. In other words that the treatment causes the outcome (the effect)" [WRH+00, p.64].

**Construct validity** "This validity is concerned with the relation between theory and observation. If the relationship between cause and effect is causal, we must ensure two things: 1) that the treatment reflects the construct of the cause well (see left part of Figure 4.15) and 2) that the outcome reflects the construct of the effect well (see right part of Figure 4.15)" [WRH+00, p.64] (the number of the figure has been adjusted to this thesis).

**External validity** "The external validity is concerned with generalization. If there is a causal relationship between the construct of the cause, and the effect, can the result of the study be generalized outside of the scope of our study? Is there a relation between the treatment and the outcome?" [WRH+00, p.64].
Figure 4.15: Experiment Principles (from Wohlin, [WRH+00, p.64])
Threats to validity endanger the possibilities to draw the above-described conclusions. In the following sections, I analyse the threats to validity for the conclusion, internal, construct and external validity.

### 4.3.1 Conclusion Validity

Threats to conclusion validity hinder the drawing of correct conclusions on the treatment-outcome-effect. Threats include low statistical power, violated assumptions of statistical tests, and choosing a statistical test and a significance rate just because it fits the results. Other factors such as unreliable measurements, biased treatment of the experiment groups, random disturbances during the experiment, and random heterogeneity of the experiment groups also threaten the conclusion validity [WRH].

The conclusion validity was problematic in this thesis, as a relatively small number of participants took part. Only with a strong effect, a high statistical power of results can be achieved. For information of the statistical power, cf. [Sac97, p.196].

Only hypothesis 3 was stated sufficiently quantitative in advance to conduct statistical tests. Here, the used test was chosen first and then applied, no "fishing" for the right test was done. Hypothesis 3 was tested using Welch's t-test, which assumes that the samples result from a normal distribution. This is unknown here, thus, the results need to be interpreted carefully [Sac97, p.200].

Other threats were controlled as much as possible, especially the homogeneity of the groups was improved by balancing. However, certain influences of the above-mentioned factors could probably not be fully prevented.

Overall, the observed treatment-outcome relation had to be carefully evaluated. Only strong effects allowed to draw the conclusion of having significant differences.

### 4.3.2 Internal Validity

The internal validity is the degree to which changes in the dependent variables of an experiment are indeed results of changing the independent variables. An experiment has a high internal validity if the experimenters controlled all relevant interfering variables well.

Threats to internal validity are circumstances that hinder this tracing back. Several classes of threats are named by Prechelt in [Pre01]. I present the threats applying to this study and the measures taken against them in the following. Table 4.2 summarises the threats and the measures taken.

For this experiment, I controlled the different capabilities of the students by evaluating their pre-experiment exercises and then randomly assigning the same amount of students from the better and worse half to each experiment group, as discussed in section 4.1.3. However, the fact that the groups are not randomised is a further threat to validity: Prechelt speaks of selection effects if the non-random assignment to experiment groups influences the outcome of the experiment. To avoid selection effects, the participants were only divided into two partitions, and were
Table 4.2: Threats to internal validity

<table>
<thead>
<tr>
<th>Threat</th>
<th>Measures</th>
</tr>
</thead>
</table>
| Different capabilities of the experiment groups | • Assign participants to groups based on results of preparatory exercises  
• Both groups apply both approaches (Cross-over design) |
| Maturation effects             | • Cross-over design  
• Time restriction |
| Bias                           | • Aim for neutrality of experimentators       |
| Help of experimentators        | • Protocol questions and assess influence     |

randomly assigned within the groups. Next to selection effects, the non-random assignment can be subject to regression effects. If a participant had for his or her particularly good results in the preparatory exercise, it was likely that his or her performance will decrease in the experiment (regression towards the mean, [Pre01]). However, the effect could equally be observed for the less successful group (showing better results in the experiment). As both experiment had successful and less successful participants, the effect overall neutralises itself.

A further threat identified by Prechelt is maturation effects of the participants. Firstly, participants may learn how to deal with the kind of task in the first session, and apply their knowledge in the second session, thus distorting the results. A cross-over design weakens this effect, as the participants use a different approach in the second session. However, there may be learning effects independent of the approaches and related to the general nature of both tasks. To identify such influences after the experiment, participants are asked in the qualitative questionnaires (question 4, appendix B.4.3) whether they were able to apply experiences from the first session in the second.

Tiredness is another maturation effect. Tired participants might change their way of solving a task, thus threatening the internal validity, as only the results for later design options were influenced. This effect was met by applying time restrictions. Finally, sequencing effects are present if participants can use findings of the results of the first session in the second, e.g. that they do not have to perform certain calculations again because they still know the values from the first session. With the given tasks and the cross-over design, however, such effects cannot apply, because the participants were not allowed to just guess the performance (based on findings in the first session, they might think to already know that a cache is useful) and they used a different approach in the second session (and thus were not able to apply findings like the manual calculation of delay at a passive resource for SPE).

Another issue threatening the internal validity of the experiment is the fact, that the students knew that the experimentators developed Palladio. Therefore, they might have been biased towards or against Palladio and shown a different motivation to complete the tasks for each approach. Here, the experimentators needed to show neutrality towards the approaches and try not to favour one approach.

Because the experimentators helped the participants with problems during the experiment session, they may have influenced the duration. If more help is given to the participants applying one approach, this additionally distorts the comparison of the duration. It was difficult to avoid
such effects, as some experimentators were experts for Palladio and others for SPE, so that different people answered questions for the different approaches. I could only assess the influence of this threat after the experiment. To do so, the record of questions can be used.

### 4.3.3 Construct Validity

An experiment with a high construct validity ensures that the persons and settings used in it represent the analysed constructs well. SPE and Palladio represent the construct of performance prediction methods. It might be argued that the two methods are not directly comparable, for example, because Palladio is specifically designed for component-based systems, less mature, but has more up-to-date tool support. However, both methods use design models similar to annotated UML diagrams, which are considered the de-facto standard in model-based performance prediction [BMIS04].

In order to represent performance predictions adequately, I chose to include the evaluation of different design alternatives in this experiment. The design alternatives represent well-known performance patterns [SW02] and were created after typical performance-enhancing architectural changes (e.g., caches, replication, etc.) for the domain of business information systems. To assess whether the evaluation of the design alternatives was too easy and could with the same accuracy also be estimated manually, I asked the students to rank the design alternatives after initial reading before conducting the modelling and analysis.

However, the tasks were restricted by the differences of the two approaches to ensure their comparability (cf. section 2.3.2). Thus, the chosen setting was a compromise between the target settings of the SPE and the Palladio approach and thus did not precisely reflect each approach’s target setting.

I chose students as the performance analysts in this study. Their competence is not as high as long-term experienced performance analysts in the software industry. However, all of the students had completed their undergraduate studies (and were therefore no beginners), and were given extensive training on both methods. In fact, the results might be even more meaningful as if I had used experienced performance analysts, because the students had the same initial knowledge about both methods. Whether the capability of students is comparable to software developers in industry is subject to discussion [Pre01].

Predictions of individual students might not be comparable to predictions of performance engineering teams, who analyse systems in industry. However, I argue that most smaller software companies are not able to employ full performance engineering teams, and merely use individuals, if they do performance prediction at all.

### 4.3.4 External Validity

The external validity is the degree to which the results of an experiment can be generalised to other, in particular practical, situations. In [Pre01] p.154, Prechelt states that the most important influence on the external validity is the experiment task. Even if the internal validity of an experiment is very high, it may only give findings for the particular task under study. For tasks
that differ a lot from the experiment task, no statements are possible. To achieve a high external validity, the task has to represent a large number of real problems and be not to specific. In the following, I present threats to the external validity that have been identified and discuss them.

Due to organisational and time constraints, I only analysed very small systems in this experiment. It is unknown whether the results are generalisable to larger software architectures with a substantially higher complexity. However, the time needed for the experiments was several hours, so that participants probably were not able to keep all aspects of the task in mind and had to look up details again, a feature of working on more complex systems.

It is furthermore unknown, whether the measured duration of the performance prediction in the experiment scenario scales up linearly for more complex systems. This can only be answered with further experiments. Overall, it is an inherent problem of controlled experiments that only smaller-than-real systems can be analysed, as the effort is hardly feasible for larger systems.

Additionally, not the whole process of designing a component-based system was analysed, but only a single excerpt in which certain design documents were available. The findings might not be generalisable to performance predictions that are conducted throughout the software life cycle in deeper and deeper detail. The experiment task was not split into several developer roles, which hinders the generalisation to such cases, in which Palladio might be much more advantageous.

I tried to control influences by the system under study, that are not generalisable to other tasks, by analysing two systems, namely Media Store and Web Server, so that different results can be traced back to the systems and not the methods. For effects observed for both approaches, it was more likely that they resulted from the differences of the approaches. Effects that were different for the two systems have to be more carefully looked at. However, it is mostly unknown what caused the differences.

A further threat to external validity is the way the participants work. If the participants solve the problems in a different way than experts would, the results are not generalisable. As discussed in the previous section 4.3.3 and section 4.1, I did not see the fact that the participants were students as a threat to external validity, however, this is subject to discussion. To achieve a realistic way of solving the tasks, the participants were trained beforehand. However, they also needed to be comparably motivated and stressed as in a industry setting, and they needed to understand the tasks and be able to apply their knowledge. These aspects are discussed in the following.

To achieve a sufficient motivation of the participants, a minimum performance in the experiment was required to get credit for the course. Additionally, the acceptance test might have motivated
the participants to produce good results, because only after passing the test, they were allowed to leave (see also section 4.2.1).

The participants only had limited time to complete the tasks. This made the experiment more realistic, as there is always some time pressure in industry scenarios that causes a certain stress. To be able to evaluate the needed effort, it was important that the participants could not spend as much time as wanted on the tasks, but had to complete the task within the time constraints. The acceptance tests ensured that an acceptable quality was delivered.

If participants had problems with the task description, they cannot apply their trained way to solve the tasks and the external validity is endangered. Note that this does not affect the internal validity, because the task description was almost identical for both approaches. If participants had difficulty with the tasks, the difference in the results could still be traced back to difference of the approaches, but only for this setting, i.e. that participants apply them that are not able to cope with the task at hand. The results would probably not be generalisable to other settings in which participants understand the tasks. To avoid problems with the task descriptions, I asked several graduate students who also took part in the preparation, but not in the experiment, to solve the tasks and improved the tasks based on their feedback.
5 Results

In this chapter, the results of the experiments are presented. The experimental set-up leading to these results is described in chapter 4. The resulting data of the experiment can be found in appendix C. With this data, the conclusions drawn in this chapter can be reassessed.

In section 5.1, the data is analysed with the Goal Question Metric approach (GQM), for which the metrics and question have been presented in section 3.2.2. It is essential for using GQM that the resulting data is interpreted on the basis of the beforehand stated metrics and questions.

Section 5.2 discusses the results, looking at both differences between the approaches and differences between the tasks, and draws conclusions.

5.1 Results of the Metrics

In this section, the measured data is interpreted based on the GQM plan. The structure of this section follows the four questions, each being partitioned into the presentation of the metrics. The metrics are evaluated for both approaches and for both tasks. Finally, each question is answered based on the measured metrics.

5.1.1 What is the quality of the created performance prediction models?

Metric M1.1: Relative deviation of predicted mean response times between the participants and the reference model.

The tables 5.1 and 5.1 shows the average of the predicted response time deviation as measured with metric M1.1 for Palladio and SPE, respectively.

<table>
<thead>
<tr>
<th></th>
<th>$v_0^*$</th>
<th>$v_1^*$</th>
<th>$v_2^*$</th>
<th>$v_3^*$</th>
<th>$v_4^*$</th>
<th>$v_5^*$</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP1</td>
<td>1.93%</td>
<td>0.90%</td>
<td>0.49%</td>
<td>20.08%</td>
<td>3.02%</td>
<td>1.69%</td>
<td>4.69%</td>
</tr>
<tr>
<td>UP2</td>
<td>13.21%</td>
<td>2.20%</td>
<td>4.15%</td>
<td>13.23%</td>
<td>4.42%</td>
<td>3.51%</td>
<td>6.79%</td>
</tr>
<tr>
<td>Web Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP1</td>
<td>1.00%</td>
<td>11.07%</td>
<td>1.94%</td>
<td>4.23%</td>
<td>4.55%</td>
<td>9.40%</td>
<td>5.47%</td>
</tr>
<tr>
<td>UP2</td>
<td>15.92%</td>
<td>20.35%</td>
<td>10.87%</td>
<td>10.67%</td>
<td>2.57%</td>
<td>3.64%</td>
<td>10.67%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

propDevMeanRespPal

Table 5.1: Metric M1.1 Relative deviation of the predicted response times for Palladio
I first look at the averages for the approaches. For the Media Store, the results of the participants using SPE had an approximately twice as high deviation from the reference model than the participants using Palladio. For the Web Server, the results of the participants using Palladio on \( UP1 \) had a twice as high deviation from the reference model than the participants using SPE, and a similar one for \( UP2 \). In average, the deviation was lower for Palladio.

To find the reasons for the deviation, I further look at the average deviation for each usage profile. Interestingly, the deviation varied a lot between the different design alternatives. For the Media Store and Palladio, the variant \( v_3 \) (second server), had a very high deviation, and \( v_0 \) for the \( UP2 \), too. For the Web Server and Palladio, the deviations for the \( v_2 \), the broker alternative, was very high, and for \( UP2 \), additionally the deviations for \( v_0 \), \( v_1 \) (Cache) and \( v_3 \) (Logging) were also fairly high. This led to the high average deviation of 24.52%.

For the Media Store and SPE, the deviation was high for \( v_4 \) (Reduction of the bit rate) and \( UP1 \), and exceptionally high compared to the other variants for \( v_3 \) (second server) and \( UP2 \). For the Web Server and SPE, no variant was particularly noticeable for \( UP1 \), but for \( UP2 \) the deviation of \( v_4 \) (Reduction of the bit rate) was more than 4 times the next highest deviation.

With these strong variations, the average was only limitedly meaningful. The difference between Palladio and SPE became less statistically significant, as the discriminatory power decreased [Pre01, p.232]

### Metric M1.2: Passed K-S Test ratio of predicted response time distribution and reference

Table 5.3 shows the ratio for passed K-S tests with a significance level of 0.05, as defined in metric M1.2 as \( \text{PassRatio}_{v,u,Pal} \), for the different variants, usage profiles and systems. The overall average \( \text{PassRatio}_{Pal} \) is 0.24. For about three-quarters of the predicted distributions, the null hypothesis that predicted distribution and reference distribution result from the same distribution function is rejected, because it is significantly improbable.

The p-values of the test vary greatly from (rounded) 0 to 0.97, with a median of 0, an average of 0.09 and a standard deviation of 0.21.

For this metric, it had to be taken into account that the sample sizes of the simulated data was very large, containing thousands of data points. A large sample size leads to a high power of the KS-test, but also leads to small differences causing a rejection of the null hypothesis [Sac97, p.197]. Thus, the high rejection rate here might be due to rather small differences in the actual
underlying distributions, which had a large influence on the results because of the large sample size (cf. section 5.2.1).

**Metric M1.3: Percentage of correct design decisions**

In the following, I firstly present the results for metric M1.3 for each system. I compared the results of the reference model (cf. section 4.2.2 for the Media Store and 4.2.2 for the Web Server) with the participants rankings and assessed $\text{perc}_{s,u,a}$.

**Media Store**

The exact evaluation of the metric for usage profile 1, only considering the best alternative, resulted in the following values:

- For SPE, only 2 out of 7 participants who ranked all variants identified the bit rate conversion as the best option.
- For Palladio, the one participant who ranked all variants did so.

All other participants did not model all options and their results cannot be used.

For usage model 2, the following values resulted from the participants’ rankings:

- For SPE, 7 out of 7 participants who ranked all variants identified the cache as the best option.
- For Palladio, the one participant who ranked all variants did not so, but placed both options on rank 1.

The number of participants who ranked all design options was low, especially for Palladio. Thus, I omit the relative values for $\text{perc}_{MS,u,a}$ here, as they do not have a significant meaning and may even be misleading.

Especially for the Palladio approach, there should not be a distinction between best and second-best alternative, as the graphs in the CDF laid very close to each other. If, due to their small differences, both options were ranked on first place together in a cluster, all of the above participants chose right, because all of them identified either the bit rate or the cache option as the best design option and ranked the respective other one second-best. I defined an adapted metric $\text{perc}_{cl,s,u,a}$ that measured the correct design decisions recognising the clusters. This evaluated to:

<table>
<thead>
<tr>
<th>Media Store</th>
<th>$v_i^1$</th>
<th>$v_i^2$</th>
<th>$v_i^3$</th>
<th>$v_i^4$</th>
<th>$v_i^5$</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UP_1$</td>
<td>0.71</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.33</td>
<td>0.71</td>
</tr>
<tr>
<td>$UP_2$</td>
<td>0.00</td>
<td>0.29</td>
<td>0.60</td>
<td>0.25</td>
<td>0.00</td>
<td>0.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Server</th>
<th>$v_i^1$</th>
<th>$v_i^2$</th>
<th>$v_i^3$</th>
<th>$v_i^4$</th>
<th>$v_i^5$</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UP_1$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$UP_2$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.13</td>
<td>0.29</td>
<td>0.00</td>
<td>0.67</td>
</tr>
</tbody>
</table>

| Overall average $\text{PassRatio}_{P_{al}}$ | 0.24 |

Table 5.3: Metric M1.2: Passed K-S Test ratio of predicted response time distribution and reference: $\text{PassRatio}_{v,u,P_{al}}$
\[ \text{perc}_{cl,MS,u,a} = 1, u \in UP, a \in A \]

**Web Server**  The evaluation of the metric for usage profile 1 results in the following values:

- For SPE, 8 out of 8 participants who ranked all variants identified the second server \( v_{4}^{WS} \) as the best option: \( \text{perc}_{WS,UP1,SPE} = 1 \).
- For Palladio, 4 out of 6 participants who ranked all variants did so. Of the two others, one really predicted a lower response time for the cache \( v_{1}^{WS} \), the other 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. I get \( \text{perc}_{WS,UP1,Pal} = 0.67 \).

All other participants did not model all options and their results cannot be used.

For usage model 2, the following percentages result from the participants’ rankings.

- For SPE, 7 out of 8 participants who ranked all variants identified the second server \( v_{4}^{WS} \) as the best option. The other participant predicted a lower response time for the cache alternative \( v_{1}^{WS} \). I get \( \text{perc}_{WS,UP2,SPE} = 0.88 \).
- For Palladio, 5 out of 5 participants who ranked all variants did so: \( \text{perc}_{WS,UP2,Pal} = 1 \).

**Combined**  I evaluated \( \text{perc}_a, a \in A \) for the original definition of the metric [M1.3] and as \( \text{perc}_{cl,a}, a \in A \) with the recognition of equivalent clusters for the Media Store system. 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.

\[
\begin{align*}
\text{perc}_{SPE} & = 0.8 \\
\text{perc}_{Pal} & = 0.77 \\
\text{perc}_{cl,SPE} & = 0.97 \\
\text{perc}_{cl,Pal} & = 0.85
\end{align*}
\]

In both cases, the percentage is higher for SPE.

For the estimated initial rankings, the percentage of correct design decisions can be calculated analogously. I use \( \text{perc}_{Initial,a} \) and \( \text{perc}_{Initial,cl,a}, s \in S \), that are defined analogously to \( \text{perc}_a \) and \( \text{perc}_{cl,a} \). The results are presented in table 5.4.

**Metric [M1.4]: Permutations in design decision rankings, recognising clusters**

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. There were several options to cope with this problem:
(a) Without recognising clusters \( \text{percInitial}_s \)

<table>
<thead>
<tr>
<th></th>
<th>Media Store</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>( UP1 )</td>
<td>0.33</td>
<td>0.35</td>
</tr>
<tr>
<td>( UP2 )</td>
<td>0.28</td>
<td>0.35</td>
</tr>
<tr>
<td>Average</td>
<td>0.31</td>
<td>0.35</td>
</tr>
</tbody>
</table>

(b) With recognising clusters \( \text{percInitial}_{cl,s} \)

<table>
<thead>
<tr>
<th></th>
<th>Media Store</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>( UP1 )</td>
<td>0.78</td>
<td>0.35</td>
</tr>
<tr>
<td>( UP2 )</td>
<td>0.61</td>
<td>0.35</td>
</tr>
<tr>
<td>Average</td>
<td>0.69</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Table 5.4: Metric \( \text{M1.3} \): Average percentage of the correct design decisions of the initially estimated design options rankings

- First, the incomplete rankings as given by the participants could be used, just omitting the ranks that were not used. However, this distorted the results because a participant who did not assess the best design option, but ranked the rest of them relatively correctly, has a number of ranked alternatives permutations, because he ranked the second best alternative best, and so on and so forth.

- The second alternative was to adjust the participants ranking to the actual positions. For the example above, one could assume that the participants ranked the options on rank 2, 3 and 4. The actual ranking of the participant would not be changed, just the used numerical ranks. However, this alternative would be advantageous for an approach that has less complete rankings. A participant that just ranked three out of four options would have a higher probability of just guessing right.

- The third alternative was to discard all incomplete rankings. However, this would result in less data points.

The distortion caused by the second option could be reduced by weighting the relative permutation score \( \text{propPerm}_{s,u,p} \) by the number of ranks assessed in the average calculation. Thus, complete rankings had a higher influence on the overall metric \( \text{propPerm}_a \), based on the number of ranks assessed. In so doing, I expected to balance the incompleteness of the ranking. Thus, as option 2 provided more data points than option 3 and was expected to result in lower distortions than option 1, I used option 2 to cope with the missing ranks.

Let \( V^s_p \subseteq V^s \) be the set of variants participant \( p \in P \) defined a ranking for. I redefined metric \( \text{M1.4} \) as given below. This redefinition did not change the outcome if all participants assigned all ranks in their ranking.

\[
\text{Metric } \text{M1.4 } \text{propPerm}_a = \frac{\sum_{s \in S, p \in P, s,u \in UP} \text{propPerm}_{s,u,p} \cdot |V^s_p|}{\sum_{s \in S, p \in P, s,u \in UP} |V^s_p|}
\]
This metric furthermore needed a clustering of the predicted response time for the reference. The predicted response times of the reference solutions for the Media Store system are depicted in the figures 4.7(a), 4.8, 4.7(b) and 4.9. The participants were asked to rank the variants \( v_{MS}^{1} \) to \( v_{MS}^{4} \). The respective response times for the Web Server system are depicted in the figures 4.12(a), 4.13, 4.12(b) and 4.14. Here, only the alternatives \( v_{WS}^{1} \) and \( v_{WS}^{3} \) to \( v_{WS}^{5} \) should be ranked. The respective missing variant was the broker alternative, which was not designed to improve the performance and thus was not ranked.

These results were clustered as given in table 5.5 for the Media Store system and in table 5.6 for the Web Server system.

As mentioned above, the best and second best option, i.e. the bit rate conversion \( v_{MS}^{4} \) and the cache component \( v_{MS}^{1} \), were almost indistinguishable for the Media Store system and both usage profiles with the Palladio approach. Thus, they belonged to one class in the clustering. For usage profile 1, the pool \( v_{MS}^{2} \) and second server \( v_{MS}^{3} \) options had sufficiently different response times and were assigned to two separate classes for both approaches. For usage profile 2, these two options had very similar response times for both approaches and were clustered in one class. For the SPE approach, the best and second best option for usage profile 1 can be distinguished. However, having more classes for the SPE approach than for the Palladio approach distorts the results. Thus, the best and second best option were also clustered for SPE.

For the Web Server system and usage profile 1, the pool \( v_{WS}^{5} \) and the logging \( v_{WS}^{3} \) option were similar in respect to the predicted response time for both approaches, thus they formed a class. The two best options were different, each forming its own class. For usage profile 2 all options could be readily distinguished from each other for both approaches, thus, they all formed a class.

Having defined the clusters, I determined the values for metric M1.4. Table 5.7 shows the results as well as intermediate step of the calculation for the different systems and usage profiles. For Palladio, the ranks were wrong by 6.5% of the maximum possible permutation. For SPE, the
Table 5.7: Metric M1.4 (redefined): Weighted average of the permutation score of the design options rankings

<table>
<thead>
<tr>
<th>propPerm</th>
<th>Media Store</th>
<th>Web Server</th>
<th>Weighted average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palladio</td>
<td>0 0 0.11 0.09</td>
<td>0.065</td>
<td></td>
</tr>
<tr>
<td>SPE</td>
<td>0.04 0.1 0.08 0.06</td>
<td>0.073</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.8: Metric M1.4: Average of the permutation score of the initially estimated design options rankings

<table>
<thead>
<tr>
<th>For estimated ranking</th>
<th>Media Store</th>
<th>Web Server</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP1</td>
<td>0.32</td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>UP2</td>
<td>0.25</td>
<td>0.34</td>
<td>0.30</td>
</tr>
<tr>
<td>Average</td>
<td>0.29</td>
<td>0.34</td>
<td>0.31</td>
</tr>
</tbody>
</table>

Evaluation of hypothesis 1: With both approaches, the created models are similar to the reference model

With both approaches, the predictions of the participants only deviate in average 6.9% and 8.3% from the predictions of the reference model for Palladio and SPE, respectively. I considered this to be similar to the reference model and to indicate that the hypothesis 1 is true. However, for single variants, the deviation was much higher (see tables 5.1 and 5.2). These are unacceptable high and pose a threat to hypothesis 1. Still, overall, hypothesis 1 is not invalidated.

5.1.2 What are the reasons for the model’s quality?

Metric M2.1: Number of problems and classification Problems were documented via the question protocol, the protocol of the acceptance test and a check of the final models.

The class of problems with the approaches themselves were further refined to group the different aspects of the approaches that led to problems. Tool problems were classified as either being a problem with usage of the tool, being a problem with interpreting error messages or being a tool bug.

A problem on the question protocol was classified as being major, if the participant did not know an essential concept of the approach or the tool or how to handle something not directly supported by the tool, if a major error was detected due to help with an error message of the
system, or if there was a major bug in a tool that could not have been found without deep knowledge of the tool or that could not be explained. In short, all major problems would have resulted in major errors greatly distorting the predictions or preventing the modelling at all. A problem was classified as being minor if it was an error that did not or barely influence the performance, or if it was a question of a participant asking for the right out of some alternatives, all of them leading to minor errors only. All other problems were classified as being intermediate.

In the acceptance test, an error was usually detected because it caused a prediction to be out of the expected bounds. As there was a certain tolerance on when a prediction lies without these bounds, the detected error could usually be classified as being major or intermediate, because it distorted the performance prediction (and was detected). However, while checking the models for the cause of a distortion, several other errors may be found, which could be both classified as intermediate and minor. It was classified as minor if it did not or barely influence the performance, otherwise, it was classified as intermediate.

In the resulting models, all major problems should already have been corrected because of the acceptance test. However, in three cases, the reason for the distorted prediction could not be found in the acceptance test, but was discovered when checking the model afterwards. Thus, three major errors remained in the models. The model for the original system was checked in detail, for all other models the changed parts were checked.

The absolute number of problems for the three dimension occurrence, gravity and problem area can be found in the appendix C.3. Here, I present relative, cumulated values.

First, the problem areas are presented, cumulating the values on the occurrence dimension to identify problematic areas of methodology and tools. I kept the severity dimension to not mix up grave and minor problems and draw wrong conclusions as a result. Tables 5.9 to 5.11 show the relative number of problems for the different areas, for the Media Store, the Web Server, and for both. After that, I analysed at what point problems actually occurred (i.e. looked at the occurrence dimension).

Table 5.9 shows the problem areas with the task itself, for both approaches. Participants had problems related to the control flow, e.g. they asked how the control flow was for the bit rate conversion ($v_{4}^{MS}$) or were corrected in the acceptance test because the their cache ($v_{1}^{MS}$) was queried once and not per file. A second problem field was the performance annotation. A common error here was to forget to annotate the demand to create a second thread for the paralleled logging option ($v_{3}^{WS}$), which was often detected when checking the models. Finally, participants had problems with the description of the usage profile, e.g. they asked how to interpret the distribution of the file sizes. Most problems were related to annotations, but they were mostly minor or intermediate. The most major problems were related to the control flow. In average, every participant had more than one problem with the task description, which is visible in the last column of table 5.9 in the respective overall sum. The least problematic areas of the three presented was the usage profile area. Interestingly, these observations are made for both system, Media Store and Web Server.

Table 5.10 shows the problems in the different areas for Palladio, and table 5.11 for SPE. For both used tools, I identified the problem areas of tool usage, of interpreting the error messages and of bugs of the tool. Here, much more problems occurred with Palladio (in average 2.27 per
participant) than with SPE (in average 0.24 per participant). Although the number of participants was relatively small, and outliers might strongly have influenced this result, I 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. 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 much more usage problems with the Web Server task than with the Media Store task. For SPE, there were some problems with the usage of the system, e.g. where to specify global parameters, and some very few with bugs.

For each methodology, different areas were identified. For Palladio, these areas were the usage of parameters, especially the usage of component parameters, the handling of data types and annotation units, the assembly and the usage model. Here, in average most problems concerned the specification of parameters, followed by the specification of component parameters and of types and units. Interestingly, this relation is very pronounced for the Media Store, but different for the Web Server, where there were equally many problems with parameters and types and units, followed by component parameters. Overall, the participants using Palladio for the Web Server task had more problem with the tool that 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.

For SPE, these areas were the specification of the overhead matrix, the handling of the distributed system, the modelling of passive resources (thread and connection pool), the handling of scenarios and projects, miscalculations, the handling of probabilities and the solution of the model. Most problems concerned the overhead matrix, e.g. creating a wrong mapping between software and hardware resource requirements. The next most often problem areas are the handling of distributed systems and the handling of passive resources. The handling of prob-

<table>
<thead>
<tr>
<th>Severity</th>
<th>Control flow</th>
<th>Annotation</th>
<th>Usage Profile</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0.06</td>
<td>0.31</td>
<td>0</td>
<td>0.38</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.44</td>
<td>0.06</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td>major</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>Sum</td>
<td>0.69</td>
<td>0.38</td>
<td>0.25</td>
<td>1.31</td>
</tr>
<tr>
<td>Web Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0</td>
<td>0.19</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.19</td>
<td>0.88</td>
<td>0.06</td>
<td>1.13</td>
</tr>
<tr>
<td>major</td>
<td>0.13</td>
<td>0.06</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>Sum</td>
<td>0.31</td>
<td>1.13</td>
<td>0.06</td>
<td>1.50</td>
</tr>
<tr>
<td>Both systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0.03</td>
<td>0.25</td>
<td>0</td>
<td>0.28</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.31</td>
<td>0.47</td>
<td>0.16</td>
<td>0.94</td>
</tr>
<tr>
<td>major</td>
<td>0.16</td>
<td>0.03</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>Sum</td>
<td>0.50</td>
<td>0.75</td>
<td>0.16</td>
<td>1.41</td>
</tr>
</tbody>
</table>

Table 5.9: Metric M2.1: Relative number of task related problems
<table>
<thead>
<tr>
<th>Tool</th>
<th>Methodology</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Usage</td>
<td>Error</td>
<td>Bug</td>
<td>Sum</td>
<td>Parameters</td>
<td>Component parameters</td>
<td>Types and units</td>
<td>Assembly</td>
<td>Usage model</td>
<td>Sum</td>
</tr>
<tr>
<td>Media Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.43</td>
<td>0.14</td>
<td>0.00</td>
<td>0.00</td>
<td>0.57</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.43</td>
<td>0.43</td>
<td>0.14</td>
<td>1.00</td>
<td>1.00</td>
<td>0.29</td>
<td>0.57</td>
<td>0.29</td>
<td>0.00</td>
<td>1.86</td>
</tr>
<tr>
<td>major</td>
<td>0.14</td>
<td>0.00</td>
<td>0.14</td>
<td>0.29</td>
<td>0.57</td>
<td>0.29</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.86</td>
</tr>
<tr>
<td>Sum</td>
<td>0.57</td>
<td>0.43</td>
<td>0.29</td>
<td>1.29</td>
<td>1.57</td>
<td>1.00</td>
<td>0.71</td>
<td>0.00</td>
<td>0.00</td>
<td>3.29</td>
</tr>
<tr>
<td>Web Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0.25</td>
<td>0.00</td>
<td>0.25</td>
<td>0.50</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.25</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.88</td>
<td>0.38</td>
<td>0.13</td>
<td>1.38</td>
<td>0.63</td>
<td>0.13</td>
<td>0.38</td>
<td>0.00</td>
<td>0.00</td>
<td>1.13</td>
</tr>
<tr>
<td>major</td>
<td>1.13</td>
<td>0.00</td>
<td>0.25</td>
<td>1.38</td>
<td>0.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.13</td>
<td>0.13</td>
<td>0.50</td>
</tr>
<tr>
<td>Sum</td>
<td>2.25</td>
<td>0.38</td>
<td>0.63</td>
<td>3.25</td>
<td>0.63</td>
<td>0.38</td>
<td>0.63</td>
<td>0.13</td>
<td>0.13</td>
<td>1.88</td>
</tr>
<tr>
<td>Both systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td>0.13</td>
<td>0.00</td>
<td>0.13</td>
<td>0.25</td>
<td>0.00</td>
<td>0.34</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.41</td>
</tr>
<tr>
<td>intermediate</td>
<td>0.65</td>
<td>0.40</td>
<td>0.13</td>
<td>1.19</td>
<td>0.81</td>
<td>0.21</td>
<td>0.47</td>
<td>0.00</td>
<td>0.00</td>
<td>1.49</td>
</tr>
<tr>
<td>major</td>
<td>0.63</td>
<td>0.00</td>
<td>0.20</td>
<td>0.83</td>
<td>0.29</td>
<td>0.14</td>
<td>0.13</td>
<td>0.06</td>
<td>0.06</td>
<td>0.68</td>
</tr>
<tr>
<td>Sum</td>
<td>1.41</td>
<td>0.40</td>
<td>0.46</td>
<td>2.27</td>
<td>1.10</td>
<td>0.69</td>
<td>0.67</td>
<td>0.06</td>
<td>0.06</td>
<td>2.58</td>
</tr>
</tbody>
</table>

Table 5.10: Metric \textbf{M2.1} Relative number of Palladio related problems
<table>
<thead>
<tr>
<th>Tool</th>
<th>Methodology</th>
<th>Usage</th>
<th>Error</th>
<th>Bug</th>
<th>Sum</th>
<th>Overhead</th>
<th>Distributed system</th>
<th>Projects</th>
<th>Passive Resource</th>
<th>Miscalculation</th>
<th>Probabilities</th>
<th>Solution</th>
<th>Sum</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.11</td>
<td>0.11</td>
<td>0.44</td>
<td>0.00</td>
<td>0.11</td>
<td>0.56</td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
<td>1.22</td>
<td>1.33</td>
</tr>
<tr>
<td>intermediate</td>
<td></td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
<td>0.11</td>
<td>0.78</td>
<td>0.44</td>
<td>0.00</td>
<td>0.00</td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
<td>1.33</td>
<td>1.44</td>
</tr>
<tr>
<td>major</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.78</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.11</td>
<td>1.11</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>0.11</td>
<td>0.00</td>
<td>0.11</td>
<td>0.22</td>
<td>2.00</td>
<td>0.78</td>
<td>0.11</td>
<td>0.56</td>
<td>0.22</td>
<td>0.00</td>
<td>0.00</td>
<td>3.67</td>
<td>3.89</td>
</tr>
<tr>
<td>Web Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.63</td>
<td>0.13</td>
<td>0.00</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>intermediate</td>
<td></td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.13</td>
<td>1.88</td>
<td>0.75</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.38</td>
<td>0.13</td>
<td>3.13</td>
<td>3.25</td>
</tr>
<tr>
<td>major</td>
<td></td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.13</td>
<td>0.13</td>
<td>0.00</td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.75</td>
<td>0.88</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>0.25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.25</td>
<td>2.00</td>
<td>0.75</td>
<td>0.13</td>
<td>0.63</td>
<td>0.13</td>
<td>1.00</td>
<td>0.13</td>
<td>4.75</td>
<td>5.00</td>
</tr>
<tr>
<td>Both systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minor</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
<td>0.06</td>
<td>0.22</td>
<td>0.00</td>
<td>0.06</td>
<td>0.59</td>
<td>0.12</td>
<td>0.06</td>
<td>0.00</td>
<td>1.05</td>
<td>1.10</td>
</tr>
<tr>
<td>intermediate</td>
<td></td>
<td>0.12</td>
<td>0.00</td>
<td>0.00</td>
<td>0.12</td>
<td>1.33</td>
<td>0.60</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.19</td>
<td>2.23</td>
<td>2.35</td>
</tr>
<tr>
<td>major</td>
<td></td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
<td>0.45</td>
<td>0.17</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>0.93</td>
<td>0.99</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>0.18</td>
<td>0.00</td>
<td>0.06</td>
<td>0.24</td>
<td>2.00</td>
<td>0.76</td>
<td>0.12</td>
<td>0.59</td>
<td>0.17</td>
<td>0.50</td>
<td>0.06</td>
<td>4.21</td>
<td>4.44</td>
</tr>
</tbody>
</table>

Table 5.11: Metric [M2.1] Relative number of SPE related problems
abilities was irrelevant for the Media Store task, but created the more problems for the Web Server task. The proportion of the problems stays approximately the same for both systems (except for the probabilities, see above). Overall, participants using SPE had 4.21 methodology problems per participant in average.

Next, I look at the occurrence dimension, the results are shown in table 5.12. 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.

<table>
<thead>
<tr>
<th></th>
<th>Palladio</th>
<th>SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Methodology</td>
<td>Tool</td>
</tr>
<tr>
<td>Question protocol</td>
<td>0.38</td>
<td>2.13</td>
</tr>
<tr>
<td>Acceptance test</td>
<td>0.31</td>
<td>0.06</td>
</tr>
<tr>
<td>Error in models</td>
<td>0.75</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 5.12: Metric M2.1 Relative number of problems separated by occurrence

Subquestion 2.1: Are the approaches comprehensible?

From metric M2.1: Problems related to comprehension To assess the comprehensibility, metric M2.1 is used as well. The methodology area contains problems based on missing understanding of the approaches. Only the miscalculation area of SPE is questionable here, because miscalculations may also result from an oversight. I still count these problems for the comprehensibility, because a very well understood approach likely results in less oversights. In so doing, table 5.10 and 5.11 already contain the results for comprehensibility in the sum of the methodology column.

Metric M2.2: Average number of times of rejection before acceptance level is reached Table 5.13 shows the average number of rejections for a participant in the acceptance test. For Palladio, a solution was rejected 0.07 times in average. For SPE, a solution was rejected 0.15 times, i.e. about twice as often. Absolutely, 11 Palladio models were rejected and 28 SPE models. Note however the limited value of the absolute numbers, as the number of checked models differed.

<table>
<thead>
<tr>
<th></th>
<th>SPE-ED</th>
<th>Palladio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Store</td>
<td>0.13</td>
<td>0.04</td>
</tr>
<tr>
<td>Web Server</td>
<td>0.17</td>
<td>0.10</td>
</tr>
<tr>
<td>Average</td>
<td>0.15</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 5.13: M2.2 Average number of rejection before acceptance level is reached
**Metric M2.3: Average number of errors in interpreting results** This metric could only be measured for solutions by the participants that included rankings of the design alternatives. Thus, like metric M1.4, this metric is adapted because not all participants had complete rankings. The adaptation was equivalent to the adaptation of metric M1.4.

With the adapted metric, I got the results shown in table 5.14. For the Media Store, no interpretation errors were made, i.e. the permutation score was 0. For the Web Server, the score was 0.06 for the SPE rankings and 0.07 for the Palladio rankings. The overall score was 0.03 for SPE and 0.05 for Palladio. Note, that less participants finished the Media Store task using Palladio, this is why the low score for the Media Store had a smaller impact on the overall score.

<table>
<thead>
<tr>
<th></th>
<th>SPE-ED</th>
<th>Palladio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Store</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Web Server</td>
<td>0.06</td>
<td>0.07</td>
</tr>
<tr>
<td>Average</td>
<td>0.03</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 5.14: M2.3 Average score for errors in interpreting results

**Metric M2.4: Subjective evaluation of comprehensibility by the participants** 19 participants answered the questionnaire. Some, however, omitted questions, so that for some of the questions, less than 19 answers had been analysed.

First, I asked for the comprehensibility of the process of each approach (questions 5 and 18). For Palladio, 17 of 18 participants stated that the process was comprehensible. One felt overwhelmed by the complexity of the PCM in Eclipse and stated that he needed more information on the role model. For SPE, 16 of 18 participants stated that the process was comprehensible. The other two participants stated that the comprehensibility was limited.

All 19 participants attributed good comprehensibility to the Palladio meta model (question 6).

The results of the evaluation of the comprehensibility of the different concepts are shown on table 5.15 for Palladio (question 7) and table 5.16 for SPE (question 19). The average grade is given as well as the standard deviation to the latter to assess the variability of the results. Recall that the scale for the grading ranged from ++ (i.e 2) to - - (i.e. -2).

Most Palladio concepts received an average grade higher than 1, with a comparably low standard deviation of less than 1. There were two exceptions: The resource environment had an average grade higher than 1, too, but here the standard deviation was higher than 1. This was because two participants graded the comprehensibility of the resource environment as lower than 0. The parametrisation had the lowest grade, thus was assessed to be hardest to understand. Here, participants mostly graded a 1 or 0, with some participants giving a 2 or -1.

The SPE concepts received lower grades. Only the software model had an averagely good grade, with a low variability. Most concepts were graded between 1 and 0, with a rather high variability of more than 1. Here, the evaluations of the participants were very different, mostly ranging from 2 to -2 and covering all grades. The concepts with the lowest subjective comprehensibility
### Table 5.15: Subjective evaluation of the comprehensibility of the Palladio concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Average grade</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository model</td>
<td>1.84</td>
<td>0.37</td>
</tr>
<tr>
<td>SEFF specification</td>
<td>1.74</td>
<td>0.45</td>
</tr>
<tr>
<td>System</td>
<td>1.61</td>
<td>0.50</td>
</tr>
<tr>
<td>Allocation</td>
<td>1.53</td>
<td>0.61</td>
</tr>
<tr>
<td>Resource environment</td>
<td>1.21</td>
<td>1.13</td>
</tr>
<tr>
<td>Usage Model</td>
<td>1.58</td>
<td>0.51</td>
</tr>
<tr>
<td>Parametrisation</td>
<td>0.58</td>
<td>1.02</td>
</tr>
<tr>
<td>Visualisation of the results</td>
<td>1.32</td>
<td>0.58</td>
</tr>
<tr>
<td>Distributions</td>
<td>1.32</td>
<td>0.48</td>
</tr>
</tbody>
</table>

### Table 5.16: Subjective evaluation of the comprehensibility of the SPE concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Average grade</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Division in scenarios</td>
<td>0.63</td>
<td>1.30</td>
</tr>
<tr>
<td>Software model</td>
<td>1.11</td>
<td>0.76</td>
</tr>
<tr>
<td>Overhead matrix</td>
<td>0.42</td>
<td>1.26</td>
</tr>
<tr>
<td>System model</td>
<td>0.63</td>
<td>1.16</td>
</tr>
<tr>
<td>Distributed systems</td>
<td>-0.53</td>
<td>1.26</td>
</tr>
<tr>
<td>Different solutions</td>
<td>0.00</td>
<td>1.14</td>
</tr>
<tr>
<td>Visualisation of the results</td>
<td>0.16</td>
<td>1.34</td>
</tr>
</tbody>
</table>
were the different solutions of SPE and the way to handle distributed systems. The latter was the only concept with an average grade of less than 0.

The help of the Palladio role model for the comprehensibility was assessed in question 8. Here, 16 participants stated that the role model helped to understand the procedure model, 2 stated that it did not. Thus, 0.89% say that it helped.

Finally, I asked the participants in question 30 which approach was easier to understand. Here, 12 participants named Palladio and only 4 SPE. Although more problems were overall present for Palladio, this fitted the result that the SPE methodology itself (in contrast to the tool) resulted in more problems than the Palladio methodology.

From metric [M2.1]: Problems with the specifications of distribution functions
For the Media Store system, no errors in the specification of distribution functions were visible in acceptance tests, question protocol and final models. For the Web server system, two rejections in the acceptance test were (among other things) due to wrong specifications of distribution functions. In both cases, the students specified a probability mass function instead of a probability density function. No questions were asked concerning the specification of the distribution function and no such problems were found.

Metric [M2.5]: Subjective evaluation of distribution functions by participants
I asked the participant whether the interpretation of the resulting distributions was harder than the interpretation of the mean value of SPE (question 22) and whether the resulting distributions were a better foundation for design decisions (question 23).

Only 4 participants stated that the interpretation of the resulting distributions was harder than the interpretation of the mean value of SPE, whereas 15 denied this. All 18 participants who answered the second questions stated that the resulting distributions were a better foundation for design decisions.

From metric [M2.1]: Problems with specifying parametrisations
The amount of problems with specifying parametrisations can be directly taken from table 5.10. It became visible that the specification and handling of parameters was the main problem source for the Palladio methodology.

Metric [M2.6]: Subjective evaluation of parametrisation by participants
First, I asked the participants to evaluate the parametrisation and name advantages and disadvantages (question 9). Here, 16 advantages and 17 disadvantages were named, partially being the same. Note that the participants did not only evaluate the comprehensibility of the parametrisation, but all aspects.

Named advantages were the explicit modelling of the usage profile and the dependencies, the flexibility, variability and reusability, as well as the more intuitive modelling.

Named disadvantages were that it was time-consuming, that the specification seemed partially redundant because everything needed to be specified (no automatisms), that the spreading of
information hindered an overview, that component parameters were too hidden and their distinction to parametrisation was unclear, that one needed much effort to hand over parameters, and that the resulting models were less maintainable.

Note that these advantages were partly named by multiple participants, and partly by one single participant. There was a contradiction between the advantage of reusability, variability and flexibility and the disadvantage of the models being less maintainable. Here, the participants had differing opinions. However, this aspect is less a subjective issue than a objectively assessable one. It could be answered with further experiments, in which the reuse of parametrised models could be tested.

Next, I asked the participants to estimate further advantages and disadvantages of the parametrisation for larger and more complex systems (question 10). Here, only one participant named an advantage, whereas 10 named a disadvantage. The named advantage was the easier change of existing models. Named disadvantages were the higher effort for the manual parameter passing (6 participants) and the resulting lack of clarity (4 participants), partly caused by the lack of naming conventions for parameters (3 of the 4 participants).

Additionally, I asked whether the parametrisation eased or hindered the specification of complex branch probabilities, as needed for the bit rate conversion design option of the Media Store system and the initial system of the Web Server (question 12). Here, 11 participants stated that the parametrisation eased the specification of complex branch probabilities, and 5 participants stated that it hindered this specification.

Finally, I asked whether potential problems with the parametrisation were due to the concept itself or rather due to the specific concrete presentation in the tool (question 16). Here, 3 participants stated that it was a problem of the concept, and 7 that it was a problem of the concrete presentation in the tools. 1 additional participant named both. 5 other participants stated that the parametrisation was not problematic.

Subquestion 2.2: Are the tools usable?

From metric [M2.1]: Problems related to the usability Again, the amount of problems with the usability of the tools can be directly taken from the tables [5.10] and [5.11]. In average, Palladio participants had 1.41 problems with the usability of the tools, thereof 0.57 during the Media Store task and 2.25 during the Web Server task. The main problem source here was the trouble of finding certain elements in the tool, and most problems resulted in questions and were documented in the question protocol. 5 participants, for example, did not know where to specify component parameters, other problematic areas were the usage profile and the composite diagram for the system. The participants using SPE had only 0.18 problems per participant in average, thereof 0.11 during the Media Store task and 0.25 during the Web Server task. Here, three problems occurred: First, a participant did not know where global parameters are specified and asked. Second, a participant wrongly copied an execution graph which was detected in the acceptance test. Third, a participant entered the demand for the encoding with a comma instead of a dot for a floating point number, which increased the demand by factor 10.
CHAPTER 5. RESULTS

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system can be modelled fairly accurately</td>
<td>Complex</td>
</tr>
<tr>
<td>The system can be better imagined</td>
<td>High initial modelling effort</td>
</tr>
<tr>
<td>Design alternatives only need to be modelled once</td>
<td>High overall modelling effort</td>
</tr>
<tr>
<td>High flexibility</td>
<td>Obscure</td>
</tr>
<tr>
<td>Eclipse plugin</td>
<td>Many bugs</td>
</tr>
<tr>
<td>Good graphical modelling</td>
<td>Complicated passing of variables</td>
</tr>
<tr>
<td>User friendly</td>
<td>Confusing auto completion</td>
</tr>
<tr>
<td>Intuitive modelling using SEFFs</td>
<td>Poor usability</td>
</tr>
<tr>
<td>Powerful because of reuse possibilities</td>
<td></td>
</tr>
<tr>
<td>Depicts clearly</td>
<td></td>
</tr>
<tr>
<td>Good for larger models</td>
<td></td>
</tr>
<tr>
<td>Clean modelling</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.17: Metric M2.7: Subjective advantages and disadvantages of the PCM Bench

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to use</td>
<td>Inflexible</td>
</tr>
<tr>
<td>Fast initial modelling</td>
<td>Less systematic</td>
</tr>
<tr>
<td></td>
<td>Imprecise</td>
</tr>
<tr>
<td></td>
<td>Simulation leads to strange results</td>
</tr>
<tr>
<td></td>
<td>GUI is old-fashioned</td>
</tr>
<tr>
<td></td>
<td>Bugs</td>
</tr>
<tr>
<td></td>
<td>Only for small designs</td>
</tr>
<tr>
<td></td>
<td>Hard to understand modelling of design alternatives</td>
</tr>
</tbody>
</table>

Table 5.18: Metric M2.7: Subjective advantages and disadvantages of the SPE-ED tool

**Metric M2.7: Subjective evaluation of usability by participants**  In the evaluation of the suitability of the tools, some participants only named advantages and disadvantages without giving an overall opinion. In the following, the amount missing to a sum of 19 are the amount of participants not giving an opinion.

14 participants stated that the PCM Bench was suited for predicting performance, none stated the opposite. 20 advantages and 18 disadvantages were named. Table 5.17 presents some, combining similar ones and omitting ones clearly related to the methodology, e.g. that the approach is component-based.

For SPE-ED, 7 stated that the tool was suited, 3 stated that it was suited with limitations, and 2 stated it was unsuited. 9 advantages and 14 disadvantages were named. 7 of the named advantages referred to the possibility to fast model the system. Table 5.18 presents some, combining similar ones and omitting ones clearly related to the methodology, e.g. that the approach only supports mean value analyses.

When asked to compare the tools (question 32 in qualitative questionnaire, see appendix B.4.3), 10 participant stated that they preferred to work with the PCM Bench, whereas 4 stated that they preferred to work with the SPE-ED tool. The reasons for the participants favouring SPE
were the faster predictions (3 participants) and easier use (1 participant). For Palladio, the participants named the higher accuracy of the models (3 participants), higher user-friendliness (4 participants), that it was more intuitive (1 participant), and more trust (1 participant).

Finally, I asked whether it would be helpful to add a textual concrete syntax for the PCM, e.g. a kind of pseudo code for the SEFFs, for some model parts and if yes, which model parts should be changed (question 15). Here only 5 participants agreed that a textual concrete syntax would be helpful, and named the SEFF (2 participants), the parametrisation (1), types and signature (1), and the allocation diagram (1).

**Subquestion 2.3: What are further reasons?**

To answer this question, I looked at the questions specifically asked to detect further reasons and analysed the answers given for other questions.

I asked for suggestions how to improve both the approaches and the tools. For Palladio (question 17), the answers are given in appendix [C.4.1](#) for SPE (question 21) in appendix [C.4.2](#), both in their original German phrasing.

For Palladio, all suggestions for improvement concern the tool, like a better support for the specification of parameters, better auto completion, a textual concrete syntax, better copy & paste possibilities, bug fixing, and more documentation. These issues were already covered in other metrics.

For SPE, most suggestions for improvement concern the tool, too, like usability in general, a more modern GUI and a better support for distributed systems. However, there were suggestions not related to the tool, namely less abstraction from the software and not only conducting mean value analyses.

Furthermore, I asked whether the participants have more trust in the predictions of one approach, if yes, which approach and why in question 29. Here, 16 participants stated that they had more trust in Palladio, 1 stated that he had the same trust in both and 2 stated that they trusted neither of the approaches. The answers in German to the why question are given in the appendix [C.4.3](#). Here, the greater detail of Palladio, the use of distributions, and the accessible source code and generated code were named.

Additionally, I asked which approach the participants preferred and why in question 31. 15 participants named Palladio and 2 SPE. The results for the why question are shown in appendix [C.4.4](#)(German). Participants preferring Palladio did so because it was closer to reality, more modern, more powerful, more intuitive, and less frustrating, and because it was using the role concept. Participants preferring SPE did so because it was easier and less complicated than Palladio.

By analysing the answers and justifications in the qualitative questionnaire, no more reasons were be detected.
Evaluation of hypothesis 2: Some potential problems arise from a lack of understanding and tool difficulty

The number of problems detected, being in average more than 4 per participant for both approaches, show that there certainly was a significant number of problems. Still, as also minor problems were included, and as the quality of the created models was overall satisfactory, they do not invalidate the 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.

5.1.3 What is the duration of predicting the performance?

Metric [M3.1]: Average duration of a prediction  First, I evaluated metric [M3.1] for the whole experiment task, thus the duration $d_p$ includes the duration of analysing the initial system and all design alternatives. In neither session, all participants were able to finish the respective task within the extended time constraints, especially participants using Palladio. To still be able to assess the duration of the whole task, I evaluated metric 1.1 for the results of $k$ participants who finished the task only. To not favour one approach, only the results of the $k$ fastest participants from the SPE group were evaluated, too, so that for both groups, the slower participants were left out [Pre01].

Figure 5.1 shows the results of metric [M3.1] for the four combinations of approaches and systems. The number of evaluated results is $k = 3$ for the Media Store (MS) and $k = 6$ for the Web Server (WS).

To be able to get more data points, metric [M3.1] was also evaluated for the analysis of the initial system only without design alternatives. All participants (except the aforementioned excluded ones) had completed the prediction for the initial systems. Figure 5.2 shows the resulting box-and-whisker diagrams, including the time to read the exercise. Note the different x-axis scale that is used in figure 5.2 as is intended to be compared to figure 5.1.

Table 5.19 shows the average metric [M3.1] for all aforementioned combinations and lists the factors $d_{Pal} / d_{SPE}$ by which the duration of making the respective Palladio prediction was slower than making the respective SPE prediction. The average factor of 1.41 was considerably close to the expected value of 1.5. However, the variance over the different systems and scopes (initial system only, whole task) was considerably high. The standard deviations of the factors given in table 5.19 is 0.39.
Chapter 5. Results

Figure 5.1: Metric M3.1: Duration of whole task for both approaches and both systems.

Figure 5.2: Metric M3.1: Duration of the initial system only, for both approaches and both systems.
CHAPTER 5. RESULTS

**Metric [M3.2]: Time needed to solve preparatory exercises**  Of the preparatory exercises, 5 were concerned with practising the SPE approach and 5 were concerned practising the Palladio approach. The participants were asked to document how much time they spent on the preparatory exercises. In average, they spent 20h 49min for the 5 Palladio exercises and 17h 25min for the 5 SPE exercises. Because the last of the Palladio exercises took the participants 8h 52min in average, I asked those participants who needed eight hours and more for the reasons for that. They mostly mentioned problems with the tool, like error messages the reasons could not be found for and bugs.

**Metric [M3.3]: Subjective evaluation by participants on needed time and effort to learn the approaches**  I asked the participant which approach needed more effort to learn it in question 24 of the qualitative questionnaire. Here, 12 participants stated that Palladio needs more effort to be learned, 5 named SPE and 2 stated that both are equally laborious to learn.

**Evaluation of hypothesis 3: The duration for a Palladio prediction is 1.5 times higher as the duration for an SPE prediction**

For the initial system only, the duration for a Palladio prediction was in average even 1.81 times higher. For the whole task, however, the duration for a Palladio prediction was in average only 1.25 times higher. In average, the duration for a Palladio prediction was only 1.41 times higher in average as the duration for an SPE prediction.

I conducted Welch’s t-test [Wel47], which is suitable to compare two distributions that have different variances, and which is available in the R tool [Dal03]. As a significance level, I chose 0.05. Thus, the p-value must be smaller than 0.05 to invalidate the null hypothesis of the test. However, Welch’s t-test assumes that the samples result from a normal distribution, which is unknown.

To assess the power of the results, I conducted a power analysis using `power.t.test` as available in the stats package version 2.5.0 [Dal07] for the R tool version 2.5.0. Usually, a power of 0.7 minimum or better 0.8 is wanted for statistical tests (cf. [Sac97, p.198]).

For the initial system only, the null hypothesis was that duration for a Palladio prediction is 1.5 times higher as the duration for an SPE prediction, i.e. that the expected value \( \mu \) for the durations differs as follows: \( \mu_{\text{Pal}} = 1.5\mu_{\text{SPE}} \). The p-value for a two-sided test was 0.154. As this value was larger than 0.05, the null hypothesis could not be invalidated with a significance level of 0.05. Still, the small p-value suggested that the null hypothesis might be wrong. The power of the test as calculated with `power.t.test` was 0.927 for the Palladio samples and 1 for the SPE samples.

I also tested for the inequality itself, with the null hypothesis that a Palladio prediction takes more than 1.5 times longer than an SPE prediction: \( \mu_{\text{Pal}} > 1.5\mu_{\text{SPE}} \). Here, the resulting p-value was 0.923. For the alternative hypothesis that \( \mu_{\text{Pal}} < 1.5\mu_{\text{SPE}} \), the p-value was only 0.077. Thus, it is unlikely that the duration of a Palladio prediction is less than 1.5 times higher as the duration for an SPE prediction, with a significance level of more than 0.1. The corresponding one-sided power analysis yielded a power of 0.965 for the Palladio samples and 1 for the SPE samples.
For the whole system, the hypothesis $\mu_{Pal} = 1.5\mu_{SPE}$ results in a p-value of 0.009 and can be rejected. The power calculated here is 0.654 for Palladio and again 1 for SPE. Here, the power is very low for Palladio, less than 70% should usually not be accepted [Sac97, p.198].

Testing for the inequality $\mu_{Pal} < 1.5\mu_{SPE}$, the p-value is 0.996, thus the inequality is very likely. Having this result, I tested the hypothesis that the duration of a Palladio prediction is even less than the duration for an SPE prediction: $\mu_{Pal} < \mu_{SPE}$ prediction. This hypothesis can be rejected with a p-value of 0.012 at a 0.05 significance level, thus, the duration of a Palladio prediction is very likely to be higher than the duration for an SPE prediction, if not as high as 1.5 times. The corresponding one-sided power analysis yields a power of 0.775 for Palladio and 1 for SPE. Thus, the power is rather low for the small Palladio sample size, but still acceptable [Sac97, p.198].

### 5.1.4 What are the reasons for the duration?

**Metric [M4.1]: Duration of the single steps** Because not all participants finished the whole tasks, the average of the duration could not be calculated straightforward. Especially for the Media Store and Palladio, only two participants provided data for all activities for all parts of the task. Just averaging the existing data would result in longer average durations for those variants that have been worked on by all participants. As a result, I present the original raw data in appendix C.2 Here, I summarise several characteristic numbers, with changing underlying sample sizes.

First, I look at the duration of the predictions for the different variants of the system. Generally, I only look at participants who completed the predictions of all variants, because of the aforementioned reasons. I include all participants that provided the duration for the variants, even if they did not give detailed time steps for each activity.

For the Media Store and Palladio, however, participant $p_6$ only omitted the prediction of variant $v_4^{MS}$, because he had to leave after the originally scheduled time. Thus, the aforementioned reasons did not apply. The inclusion of participants not giving detailed information on the activities duration applied to participant $p_3$. The resulting box plot is shown in figure 5.3(a).

For the Media Store and SPE, participant $p_9$ did not provide durations for the later variants and his results were omitted. Participant $p_{13}$ did not complete the prediction for variant $v_4^{MS}$ and was not included. Here, the aforementioned reasons apply because he did not finish within the extended time constraints. All other participants results were included in this analysis. The resulting box plot is shown in figure 5.3(b).

For the Web Server and Palladio, the two participants $p_{11}$ and $p_{15}$ did not finish within the extended time constraints and their results were not included. For the Web Server and SPE, all results were included. The two resulting box plots are shown in figure 5.4 with Palladio on the left-hand side and SPE on the right-hand side.

Note that in this comparison, I used a different number of data point for the different combinations of system and approach. Because of this, the absolute results, i.e. the absolute durations, could not be readily compared. However, assuming that the proportion of the duration for
Figure 5.3: Metric M4.1: Duration of the prediction for the different variants of Media Store

Figure 5.4: Metric M4.1: Duration of the prediction for the different variants of Web Server
initial system and the variants did not change, I could compare the factors (duration variant $x$):(duration initial system) over the different approaches and systems.

For Palladio, the median duration of the predictions of the Media Store variants $v_1$ and $v_4$ is about a third of the median duration for the initial system, and even less for the other variants. For the Web Server, this factor was about a fifth for variant $v_1$ and variant $v_3$ and even less for the other variants.

For SPE and the Media Store, this factor was about 0.85 for variants $v_3$ and $v_4$ and less for the other variants. For the Web Server, the factor was 0.45 for variant $v_4$ and less for the other variants.

**Metric M4.2: Breakdown of the duration to activities**  I first looked at the break down of the duration as measured in metric M3.1 into the different activities for the initial system only (scope is), because it represented a creation of the models from scratch and I had more data points for it.

Reading ($ra$) was only an initial reading of the task description, all participants had to read excerpts of the task again while modelling, which was included in the modelling time. For SPE, the participants did not give a separate time for the annotation of resource demands ($rd$), but included this time into modelling of the control flow ($cf$) or into modelling of the resource environment ($re$). Each experiment task contained two usage profiles, so the duration of their modelling, searching for errors and analysing was measured separately for each usage profile and then was averaged.

Figure 5.5 shows the break down of the duration of making a prediction for the initial system, without considering the duration of the evaluation of the design alternatives (scope is). It is visible that the entire modelling, including $cf$, $rd$, $re$, and $up$, was the major activity for both approaches, as expected.

However, participants using Palladio spent much more time on searching for errors, i.e. fixing wrong or missing parameters: $dact_{Pal, err, is, MS} = 28\%$ (Media Store) and $dact_{Pal, err, is, WS} = 20\%$ (Web Server). The participants using SPE only spent $dact_{SPE, err, is, MS} = 2\%$ (Media Store) and $dact_{SPE, err, is, WS} = 6\%$ (Web Server), respectively, of their time on searching for errors. The proportion of the analyses was constant for the approaches and differed only in the system under study: For the Media Store system, participants using Palladio spent $dact_{Pal, ana, is, MS} = 11\%$ of their time in average for the analyses, participants using SPE $dact_{SPE, ana, is, MS} = 9\%$. For the Web Server, participants of both groups used in average $dact_{a, ana, is, WS} = 4\%, a \in A$ of their time for the analyses.

The duration of the whole task, i.e. modelling all design alternatives (scope wt) was also composed down to these aspects. The duration of reading $dact_{a, ra, wt}$ was relatively smaller, because it had just been queried once at the beginning of the task. The other ratios of modelling, searching for errors and analysis stayed approximately the same. As the corresponding chart for all alternatives is fairly obscure due to many measured durations, it does not depict the results very well and I omit it here.

Moreover, I identified the main time consuming activity of the different variants for the different systems and approaches by looking at the durations documented by the single participants. I list
CHAPTER 5. RESULTS

Figure 5.5: Metric M4.2: Break down of the duration for the initial system

<table>
<thead>
<tr>
<th>Palladio</th>
<th>SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$ Modelling, Searching for errors</td>
<td>Modelling</td>
</tr>
<tr>
<td>$v_1$ Modelling, Analysis</td>
<td>Modelling</td>
</tr>
<tr>
<td>$v_2$ Modelling</td>
<td>Modelling</td>
</tr>
<tr>
<td>$v_3$ Modelling, Analysis</td>
<td>Modelling, Searching for errors</td>
</tr>
<tr>
<td>$v_4$ Modelling</td>
<td>Modelling</td>
</tr>
<tr>
<td>$v_5$ Analysis</td>
<td>Modelling</td>
</tr>
</tbody>
</table>

Table 5.20: Metric M4.2: Main time consuming activities for the Media Store

the activities that participants used the most time on for each variant, and order them according to how many participants spent the most time on the activity. Table 5.20 summarises the findings for the Media Store and table 5.20 for the Web Server. Most participants spent the most time on modelling the systems. However, there are participants that spent more time on searching for errors for both approaches. For Palladio, there are also participants who spent the most time on the analysis of a system, especially for the broker alternative ($v_5$) of the Media Store, where all participants using Palladio spent the most time for the analysis.

Metric M4.3: Subjective evaluation by participants on reasons for the needed time First, I asked which approach was more time-consuming to apply and why (question 25). 16 participants named Palladio as being more time-consuming to apply, and only 3 named SPE. The given reasons were the more detailed modelling (6 participants), the initial modelling effort (2), the effort to first define the components (i.e. the RDSEFFs) (1), the higher complexity (2), the SEFF modelling (1), the more difficult use of the tool (1), and the "clicking-intensive" tool (1). Two participants naming SPE mentioned the worse reuse capabilities of the SPE models.

To assess the influence of the used tool, I asked which tool was faster to use (question 27). 3 participants named the PCM Bench and 13 the SPE-ED tool.
Table 5.21: Metric M4.2: Main time consuming activities for the Web Server

<table>
<thead>
<tr>
<th></th>
<th>Palladio</th>
<th>SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_0)</td>
<td>Modelling, Searching for errors</td>
<td>Modelling</td>
</tr>
<tr>
<td>(v_1)</td>
<td>Modelling</td>
<td>Modelling</td>
</tr>
<tr>
<td>(v_2)</td>
<td>Modelling</td>
<td>Modelling</td>
</tr>
<tr>
<td>(v_3)</td>
<td>Modelling</td>
<td>Modelling, Searching for errors</td>
</tr>
<tr>
<td>(v_4)</td>
<td>Modelling</td>
<td>Modelling</td>
</tr>
<tr>
<td>(v_5)</td>
<td>Modelling</td>
<td>Modelling UP2</td>
</tr>
</tbody>
</table>

Table 5.22: Metric M4.3: Subjective advantages and disadvantages of the automated transformation named by the participants

<table>
<thead>
<tr>
<th>Advantages (No of participants)</th>
<th>Disadvantages (No of participants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to model (5)</td>
<td>Not so precise (1)</td>
</tr>
<tr>
<td>Fast (5)</td>
<td>Loss of control (1)</td>
</tr>
<tr>
<td>Transparency during modelling (2)</td>
<td>Less flexible (3)</td>
</tr>
<tr>
<td>Elegant (1)</td>
<td>Not applicable for complex cases (1)</td>
</tr>
<tr>
<td>Not manually (1)</td>
<td>Unclear what happens (1)</td>
</tr>
<tr>
<td></td>
<td>Less practical relevance (1)</td>
</tr>
</tbody>
</table>

For Palladio, I asked whether the parametrisation eased the specification of the SEFFs or whether it was an additional effort (question 11). 5 participants stated that it actually eased the modelling, and 10 participants stated that it was an additional effort.

Furthermore, I asked the participants to assess the automated transformations available in Palladio, as used the broker lookup alternative. Again, the participants were asked to name advantages and disadvantages. Table 5.22 shows both and the number of participants stating them.

**Evaluation of hypothesis 4: The most time-consuming activity is the modelling**

The results indicate that hypothesis 4 can be held: The participants spent the largest part of their time on modelling the systems, for both the modelling of the initial system and the whole system.

**5.2 Discussion of the Results**

In the following, I discuss the results of the GQM metrics. Firstly, I look at the differences of the approaches that became visible in the results. After that, I consider the quality metrics that have been specifically asked for the Palladio approach because they had no counterpart for SPE. Finally, I discuss the differences of the two systems in the two experimental tasks, Media Store and Web Server. For the comparison of the approaches and the systems, I first look at the differences of the quality of the created models and then at the differences of the duration.


CHAPTER 5. RESULTS

5.2.1 Differences of the Approaches

Quality of the created models

With the given measures, the quality (in terms of similarity to the reference model) of the prediction models created with SPE and Palladio was similar. With SPE, more participants were able to identify the best design options (metric M1.3). On the other hand, for Palladio, there was slightly less deviation from the response time of the prototype model (metric M1.1) and participants created better total rankings (metric M1.4). The quality of the Palladio distributions as measured with the K-S test in metric M1.2 is discussed at the end of this section, although it could not be compared to SPE results.

As an acceptance test was used in the experiment design, the numbers above did not reflect the quality of the models as the participants would have created them without help. Thus, the similar quality can be partly explained with the acceptance test. To find differences of the approaches, I had to look at the problems documented during the experiment, i.e. metric M2.1. From these, I could conclude how well the models would have been created without any possibility to ask or without an acceptance test. Overall, a little more problems were encountered using Palladio with 4.85 problems per participant as opposed to 4.44 problems per participant with SPE (c.f. table 5.10 and 5.11). Most problems were rated intermediate, followed by major for Palladio and minor for SPE. Overall, I conclude that Palladio, including both method and tool, was harder to use for the participants than SPE, because it led to more problems.

Still, both approaches led to a good quality of the created models in terms of similarity to the reference models. For Palladio, the complexity of the meta-model was well hidden in the tools and did not strongly influence the comprehensibility.

Problems with the methodology only

If I exclude the influence of the tool itself, i.e. the problems related to the use of the tool, error messages and bugs, and only look at problems with the methodology itself, the picture changes. The participants using Palladio only had 2.58 problems related to the method as opposed to 4.21 for SPE. This suggests that an improvement of the Palladio tool might change the results in further experiments and that the methodology itself was even better applicable than the SPE methodology.

The qualitative evaluation also supported these findings, because it suggests the Palladio methodology having a better comprehensibility. The process of both approaches was equally well understood, as was the Palladio meta model. However, for Palladio most concepts were graded better than 1, only the comprehensibility of the parametrisation was problematic, although still better than 0. Even the use of distributions was not considered a problem, but even an advantage. For SPE, most concepts were graded between 0 and 1, with the comprehensibility of distributed systems being even worse than 0. Thus, the participants subjectively understood the Palladio concepts better than the SPE concepts. This fits the fact that most participants stated that the Palladio approach was easier to comprehend.

A new aspect found by looking for further reasons is Palladio being more “modern”. This might cause the participants to be biased towards it and be more inclined to put effort in learning it.
This might explain why there were more problems with Palladio (especially with the tool) but the participants nonetheless prefer it.

However, the qualitative evaluation by the participants must be considered warily. Especially if the participants were biased because of influence of the experimentators, the subjective evaluation could be misleading. However, the results complied with the results of the objective metric M2.1 and might thus represent a fairly unbiased opinion.

**Occurrence of the problems** Interestingly, for Palladio, most of the problems occurred during the experiment and led to questions. For SPE, most problems stayed in the final models. Note, however, that these problems were mostly minor or intermediate, because they did not affect the response time very much. Additionally, participants using SPE had twice as many rejections in the acceptance test (c.f. metric M2.2). Thus, assuming internal validity, the Palladio method and tool helped the participant to avoid errors better than the SPE tool did, where errors were only detected in an acceptance test. In a real application of the approaches, the users can find support for their problems if they notice them themselves. Thus, the Palladio problems would be noticed, but SPE problems would remain unnoticed and decrease the quality of the created models. However, it is unclear whether the internal validity was strong enough to draw this conclusion, see section 5.2.3.

Moreover, the interpretation of the results from the tools influences the quality of the prediction in terms of decisions made based on them. Here, there have been only some errors for both approaches. Thus, the interpretation was not particularly problematic. However, participants using Palladio had 66% more interpretation errors than participants using SPE. As 4 participants stated in the qualitative questionnaire that the interpretation of the resulting distribution was harder than just evaluating a mean value, this might be an influencing factor. However, the interpretation was considered unproblematic by most participants. Overall, the interpretation was harder using Palladio, but not problematic for both approaches.

**Duration for a prediction**

Using SPE, the predictions could be done faster. Using Palladio took 1.17 to 2.05 times longer, depending on the system under study and the nature of the task. The proportion was higher if only the prediction of the initial system is looked at. However, this was not a realistic setting, because a usual task in performance engineering is the comparison of several alternatives. For the evaluation of several alternatives, using Palladio only takes 1.17 or 1.32 times longer. To a certain extent, this could be explained by the reuse character of this scenario: For the prediction of design alternatives, the EGs of SPE were copied and adapted, which was faster than creating new models from scratch, but still a considerable effort. However, for Palladio, the RDSEFFs of the most components could be reused as is due to their parametrisation, and only single components, their assembly and allocation needed to be changed. Thus, the evaluation of design alternatives needed a relatively lower effort with Palladio than with SPE, if compared to the effort for modelling the initial system.

These findings also fit the results of the subjective questionnaire. Most participants stated that Palladio was more time-consuming. Partly this was said to be caused by the higher initial
modelling effort, which fits the numbers above. Partly, it was also said to be based on the higher complexity of the Palladio models, suggesting that the modelling will always be more time-consuming.

The extra time needed for making Palladio predictions could be traced back to the duration of searching for errors to a certain extent. This might be partly caused by the immaturity of the tool and the limited understandability of the error messages. Using Palladio more problems occurred during creation of the model and searching for errors before the simulation, but the number of acceptance tests after simulation was lower than with SPE. The PCM-Bench performs more inconsistency checks on the models than the SPE-ED tool, therefore predictions with the PCM-Bench seem more reliable. However, both tools still allow wrong parameter settings or wrong modelling.

Additionally, participants using Palladio spent more time for the analysis of the results. A part of this higher effort could be traced back to the used analysis method for the models: For Palladio, the simulation in the tool itself took several minutes per design alternatives, whereas the SPE analytical solution was available within seconds.

Still, the participants using SPE also needed less time to model and analyse the systems. However, in this experimental setting, not considering potential reuse, SPE was favoured, because it allowed to create the models on a higher abstraction level and thus faster. The resulting SPE models were not meant for reuse, whereas this is the case for Palladio models. Furthermore, existing components were not reused in the systems under study and no code was generated from the resulting Palladio models, which might have affected the combined effort of design and implementation.

### Duration of predicting variants

Interestingly, the relative time needed to make predictions for the different variants of the systems differed for the both approaches. Next to the initial system, which took the longest for both approaches, the variants were differently time-consuming, as visible in the figures 5.3 and 5.4.

For the Media Store and Palladio, the most time-consuming prediction were the introduction of a cache component ($v^{MS}_{1}$) and the change of the bit rate ($v^{MS}_{4}$). Here, $v^{MS}_{1}$ had a lower average duration but also a higher maximum. For SPE, the introduction of a second server ($v^{MS}_{3}$) and the change of the bit rate ($v^{MS}_{4}$) were the most time consuming, with $v^{MS}_{3}$ having a higher median and a higher maximum. Thus, the variant with the most complex control flow, i.e. the change of the bit rate $v^{MS}_{4}$ was complex for both alternatives. For Palladio, the participants needed to handle several parameters. For SPE, the probabilities and the demand for the different control flow alternatives had to be assessed.

For SPE, the allocation of some components to a second server ($v^{MS}_{3}$) was particularly time-consuming, because it involved creating new scenarios, several solutions and the inserting of the resulting values in the model of the first server. For Palladio, the cache introduction ($v^{MS}_{1}$) was also time-consuming. Here, participants needed to create a second system diagram. However, this was also the case with the introduction of a database connection pool ($v^{MS}_{2}$), which was less time-consuming to analyse. Maybe $v^{MS}_{1}$ took more time because it was the first variant to analyse, and the participants learned to handle the situation for later variants. This might indicate a threat to internal validity.
In average, the duration was smallest for variant $v_5^{MS}$, i.e. the usage of a broker for component-lookups, for the Palladio approach, followed by variant $v_3^{MS}$, which is the allocation of some components to a second server. The reason probably was that both variants only needed slight changes: For $v_5^{MS}$, only the simulation settings needed to be changed, which all participants realised. For $v_3^{MS}$, only the allocation diagram needed to be done anew. For SPE, the duration was smallest for variant $v_5^{MS}$, too, followed by variant $v_1^{MS}$, i.e. the introduction of a cache. For $v_5^{MS}$, only performance annotations were needed, and the structure of the control flow remained the same. For $v_1^{MS}$, a single branch node needed to be added to the control flow with the respective performance annotations.

For the Web Server and Palladio, the introduction of a cache for dynamic content ($v_1^{WS}$) and the paralleled logging ($v_3^{WS}$) were together most time-consuming to analyse. For both, new RDSEFFs had to be created. For the cache, the use of parameters might have been time-consuming, whereas the paralleled logging required some effort to change the control flow in the main component, the MediaStore component. For SPE, the allocation of some components to a second server ($v_4^{WS}$) was most time-consuming, followed by the introduction of a thread pool ($v_5^{WS}$) in considerable distance. Here, again the creation of several scenarios and the insertion of the values into the main scenario was time-consuming. For the thread pool, the participants had to estimate the waiting time for a thread manually, which took a considerable amount of time.

The variant with the fastest prediction for the Web Server using both approaches was the use of a broker component ($v_2^{WS}$). Again, for Palladio only the simulation settings had to be changed. For SPE, the participants needed to change some performance annotations. The absolute duration was similar, which might be caused by the longer duration of simulation for Palladio. For Palladio, the next fastest variants were the introduction of a thread pool ($v_5^{WS}$) and the allocation of some components to a second server ($v_4^{MS}$). Here, the built-in mechanisms of passive resources and allocation diagrams, respectively, could be used. For SPE, the next fastest variant were the paralleled logging ($v_3^{WS}$) and the introduction of a cache for dynamic content ($v_1^{WS}$), in that order. For the first, only performance annotations were changed, because the parallelism could not be expressed otherwise. For the second, the control flow needed to be changed.

Interestingly, the Web Server variants that had a long duration for SPE were the variants quickly analysed using Palladio and the other way around. Thus, Palladio required more effort to analyse a variant including the change of control flow, i.e. the creation of new RDSEFFs. Nevertheless, SPE requires more effort for the constructs not directly supported, as passive resources and distribution of components.

Overall, variants could be analysed relatively faster compared to the duration of the initial system with Palladio. Looking at the absolute values, the average duration for predicting performance of a variant was also smaller for Palladio. However, as different sample sizes were compared, the observed result only indicate that variants could be analysed faster with Palladio. I do not present absolute quantitative values here, as they might be misleading because of the different sample size.

Additionally, it was observed that variants with slight changes to control flow and performance annotations were analysed faster with SPE, whereas, unsurprisingly, Palladio enabled fast predictions for variant using built-in concepts like broker lookup and passive resources. Complex
control flow changes like in the bit rate variant of the Media Store ($v_4^{MS}$) were relatively time-
consuming for both approaches, compared to the other variants.

**Preparation effort** Next to the duration of an actual performance prediction, I also looked
at the time needed to learn the approaches. Here, the participants spent 20% more time on the
preparatory exercises for Palladio, which suggests that learning the Palladio approach needs
more effort. Additionally, 12 participants named Palladio as being the approach requiring more
effort to be learned, whereas only 5 named SPE.

**Palladio specific results**

Specific to the Palladio approach are the use of distribution functions, both as input and output,
and the parametrisation of the models. All three do not have a counterpart in SPE. Here, I
first look at another aspect of the quality of the created model, namely the K-S test as defined in
metric $M_{1.2}$. Then, I consider the results of the metrics for the reasons of the quality specifically
targeting Palladio specifics.

The analysis of the predicted distribution function of Palladio shows that only few distributions
passed the K-S test (metric $M_{1.2}$), and therefore that most distribution probably result from a
different distribution function than the one underlying the prototypical distribution. However,
the applicability of the K-S test could be doubted for the assessment of the quality of the models.
Looking at the graphical representation of the distributions, most are very similar. Additionally,
they often have a similar mean value (see metric $M_{1.1}$).

As mentioned before, it had to be taken into account that the sample sizes of the simulated data
was very large, containing thousands of data points. A large sample size leads to a high power of
the KS-test, but also leads to small differences causing a rejection of the null hypothesis [Sac97,
p.197]. Thus, the high rejection rate here might be due to rather small differences in the actual
underlying distributions, which had a large influence on the results because of the large sample
size. To not threaten the conclusion validity by “fishing” for a fitting test, I did not look for
another test to be applied.

Thus, a failed K-S test means that the distributions were unlikely to result from exactly the
same underlying distribution, but it does not mean that the distributions were not similar to
each other. The left-hand side of figure 5.6 shows two distributions with a p-value of $4 \cdot 10^{-5}$,
thus not passing the test. Undoubtedly, the distributions are nonetheless very similar.

Additionally, the resolution of the K-S test for low p-values is too low, assigning a p-value of 0
to distributions that look similar as well as to completely different distributions. For example,
the two distributions depicted on the right in figure 5.6 have a p-value of 0 for being from the
same underlying distribution, as well two very different distributions would have. However,
there is a certain similarity between the two distributions on the right in figure 5.6, although less
than for the distributions on the left-hand side of the figure.

Overall, the K-S test was only passed if the distributions were likely to result from exactly the
same distribution. For the distributions predicted in this experiment, this was seldom the case,
24% pass the test at a 5% significance level. Thus, 24% of the models created were probably identical to the reference model (except for naming differences and the like).

Next, I investigate the influence of the Palladio characteristics, namely distribution functions and parametrisation, on the quality of the created models. Here, the specification of the distribution functions was mostly unproblematic. Only two rejections in the acceptance test for the Web Server were partly due to a wrong specification of the distribution. Additionally, only some participants gave bad grades to the distribution concept in the qualitative questionnaire. Thus, I conclude that the participants could handle the specification of distribution functions well.

The specification of the parametrisation, however, was much more problematic. As shown in table 5.10, the parametrisation of the performance models was the main problem the participants had. Especially the specification of component parameters were problematic. As component parameters were only covered in the lecture and not in the preparatory exercises, a lack of practice might add to this result. Although the parametrisation in general was listed as being a problem of the approach, the presentation of the parametrisation in the tool also might have influenced this result. Many problems were related to the concrete realisation of the parametrisation concept rather than the concept itself. For example, participants had problems to understand how to return parameters or they had problems with specifying the parameters characterisations (cf. appendix C.3.1).

The qualitative findings support this. The parametrisation concept was by far the lowest graded concept. Also, slightly more disadvantages than advantages of this concept were named. The use of parameters was said to be time-consuming and obscured the models. It was even stated that the parametrisation made the models less maintainable, which contradicted the reuse goal of the concept. For more complex systems, the use of parameters was even estimated more
problematic. However, it is unclear whether the participants were able to distinguish the concept and the realisation in the tool. Additionally, the fact that they assessed the parametrisation as being less maintainable might result from a lack of understanding.

However, there were also many advantages of the parametrisation named, which showed that the concept was partly understood and used in its intended way. Named advantages were the greater flexibility of the models, the explicit modelling of dependencies and the more intuitive modelling. Interestingly, about 2/3 of the participants stated that the use of parametrisation helped handling complex branch probabilities, like for the bit rate reduction alternative of the Media Store and the type of content for the Web Server. This partially contradicted the opinion that parametrisation was worse for more complex systems, as these could be supposed to have more complex dependencies on each other.

As most participants stated that the problems were a problem of the presentation in the tool (i.e. the concrete syntax) and not of the concept itself, results might change if the concrete syntax for parametrisation is changed in the future and further investigations take place.

5.2.2 Differences of the Systems under Study

Next to the different results of the approaches, I observed different results for the single variants and thus for the systems under study.

Quality of the prediction models

I observed some significantly different results for the two systems for some quality metrics. Firstly, the ratio of passed K-S tests is higher for the Media Store. Especially for UP1, 55% of the distributions passed the K-S test for the Media Store, whereas 0% passed the test for the Web Server. There were not particularly more errors in the final models for the Web Server, so I assume that the reason was the task itself. However, which of the differences of the task caused the deviation is unclear.

Additionally, the ranking for UP1 and the Web Server was more error prone and let to a higher permutation score than UP1 and the Media Store. As the score was normalised, the lower number of classes for the Media Store could not be the reason. The metric was defined to cope with different numbers of classes and therefore different number of clusters. The underlying reason might be connected to the reason for the aforementioned K-S test differences, which might be a reason that generally influences the quality of the models.

A further difference was the deviation of the predicted response times, which differed a lot for the single variants of the systems and was overall higher for the Web Server. It showed that the quality of the models also depended on the particular variant under study.

Relations of tasks and approaches

The deviation of the predicted response times also differed differently for the single variants and approaches. For Palladio, partly other variants featured a high deviation than for SPE. For
example, for the Media Store, the $v_3^{MS}$ (Second server) featured the highest deviation with Palladio, and $v_4^{MS}$ (Re-encoding) the highest for SPE. For the Web Server, the $v_2^{WS}$ (Broker) featured the highest deviation with Palladio, and $v_4^{WS}$ (Second server) the highest for SPE.

Another difference became visible in the problem analysis of Palladio. Here, more methodology problems occurred for the Media Store, whereas more tool problems occurred for the Web Server. However, the number of problems was overall similar. A different, i.e. wrong, classification of the problems for the different would be an explanation. However, the reader can check the problems in the appendix C.3. Learning effects from the first experiment were possible. Participants of the second group might have put more emphasis on the methodology when preparing for the experiment because of some experiences with SPE. Other maturation effects were possible, as well as inherent differences of the tasks.

For the Web Server, there were also more problems with the SPE methodology. Here, the reason probably was the calculation of probabilities, as this was relatively time-consuming for SPE and was not present at all in the Media Store. Interestingly, only some participants from the first experiment group (using Palladio for the Media Store and SPE for the Web Server) stated that the tasks have been too hard for their level of knowledge (Media Store: 3 of 10, Web Server: 2 of 10). Participants of the other group said that both tasks were adequate. This might be an indication that the Media Store task was harder when using Palladio, causing more methodology problems, whereas the Web Server task with its probability calculation was harder when using SPE.

As the probability calculations were a result of no parametrisation in SPE, it is sensible to compare the parametrisation findings of Palladio at this point of the discussion. Interestingly, for the Web Server the parametrisation of Palladio was less difficult than for the Media Store. Less than half the number of problems occurred, and no major ones. Thus, the nature of the Web Server task seems to be supported better by the Palladio parametrisation concept than the nature of the Media Store task.

### Duration for a prediction

Two major differences could be identified: For the Web Server, both the duration of modelling the control flow and the variance of the overall duration was considerably higher for both approaches.

The higher duration of modelling the control flow can be seen in fig. 5.5 and in comparing figure 5.3(b) with figure 5.4(b). I could exclude that this results from the different number of participants looked at, because the proportions remained the same if only the four fastest participants’ results were considered. The duration of reading was lower for the Web Server, possibly explaining a part of the increased modelling time: The participants might have read the task description faster in the second session, and subsequently spent more time looking up details during the modelling. However, as the modelling increases more than the reading decreases, there was probably another influence. I suppose that the nature of the task lead to this increase in modelling demand: The initial system of the Web Server included the modelling of requests for HTML and multimedia as well as static and dynamic content. The control flow was more complex: For Palladio, the parameters had to be correctly set. For SPE, one way to
implement the probabilities for the different kinds of content given in the task description was to convert them using Bayes’ Rule, which was considerably complex.

By contrast, the initial system of the Media Store was simpler. However, the variant $v_4$, i.e. the conversion of the bit rate of uploaded files, was more complex in the Media Store, requiring the considerable use of parameters and the calculation of probabilities, respectively. This might explain why the participants needed more time for the whole Media Store task (see fig. 5.1).

Additionally, the duration for the Web Server had a higher variance. For the whole task (fig. 5.1), the higher variance could be explained with the larger number of data points: 6 participants were considered for the Web Server, for the Media Store only 3. However, for the initial system, the number of results compared was similar. Here, the higher complexity of the control flow might have led to a higher variance. Participants who quickly understood the exercise were able to complete it fast, whereas other participants might have spent more time on understanding the relations of the parameters. However, I can only speculate about the reasons. Here, a more detailed observance of the participants monitoring thinking time and modelling time could give further insights. However, such a study might be infeasible without a good tool support.

**Further differences**

The average number of rejections before the acceptance level was reached was higher for the Web Server. Here, again learning effects are a possible explanation. Possibly, the participants had the experience that they can try earlier to get through an acceptance test to faster finish the overall task. However, this is only a guess and could not extracted from the data.

A potentially significant difference was that for the Media Store, no interpretation errors occurred, whereas several occurred for the Web Server. As the score was not normalised by the number of clusters and only interpretation errors across cluster boundaries were counted, however, the Media Store system was expected to have a lower number of errors. Still, some errors are not a multiple of no errors, and other reasons might be present. As the observed effect was not very large, the differences might also just be a result of chance.

### 5.2.3 Further Assessment of the Validity

Although precautions had been taken to ensure conclusion, internal, construct, and external validity, the results were analysed for possible validity threats. For the construct validity, the initially estimated rankings of the participants were assessed. The findings for conclusion, internal and external validity are discussed in this section, too.

**Conclusion Validity**

The only statistically investigated hypothesis was hypothesis 3, as it was quantifiable. Here, the effects were strong, and a large power could be achieved in most cases. However, the power
analysis assumed that the samples result from a normal distribution, which was unknown. The possible violation of the assumption poses a further threat to conclusion validity.

**Internal validity**

One identified threat to internal validity was the different capability of the participants. Based on the number of problems that occurred, I assume that the groups were equally capable. For both groups, more problems occurred for the Web Server system, and in both cases, slightly more problems occurred using Palladio. The same is true for the number of rejected acceptance tests itself: Here, both groups needed less tries when using Palladio or analysing the Media Store than the other group. For the needed time, again no effects were visible. Here, participants needed less time for the Web Server with alternatives or less time with SPE. For the modelling of the initial system, however, there was no such simple relation. For Palladio, the Web Server system was modelled faster than the Media Store, for SPE the Media Store system was modelled faster. This might result from 1) the Web Server tended to be easier to model with Palladio and the Media Store tended to be easier to model with SPE, or 2) the group applying SPE to the Media Store and Palladio to the Web Server was faster with their predictions. As the time to model the system with its alternatives as well as the number of problems did not suggest a higher capability of this group, I assume that 1) is the reason for this result, as already mentioned in the previous section.

Maturation effects were also identified as a potential threat. Although a cross-over design was planned to avoid learning effects, there were hints for a certain learning effect in the experiment, as the students were able to complete the second session faster than the first one. The students were more familiar with the experiment setting in the second session. Indeed, 10 of 18 participants answered the respective question in the questionnaires positively.

There were no signs of tiredness towards the end of the tasks. Not more acceptance tries were needed towards the end of the session for later design decisions.

Concerning a potential bias caused by the experimentators, no effects were observed. On the contrary, I noticed that students complained about the tools of both approaches and my questionnaires showed no visible bias towards one method.

Finally, the help of experimentators is a threat to internal validity. Excluding questions related to the experiment task, the participants using Palladio asked 4 questions on average, whereas the participants using SPE only asked 1.6 questions. However, questions on the tool can probably not be affected by the help of the experimentators: If a participants was unable to solve a certain problem with the tool, he could not continue without asking and being helped until the problem is solved. Excluding the questions related to the tools, participants using Palladio asked 1.61 questions on average and participants using SPE only asked 1.29 questions. Again, this number can result from 1) the participants having more problems with Palladio or 2) the experimentators giving answers for Palladio being more willing to help participants and communicating this. Also, participants using SPE needed 1.15 acceptance test in average (including passed acceptances), whereas participants using Palladio only needed 1.07 acceptance tests. This could result from 1b) either problems being more obvious with Palladio (as the tool does more checking) or again 2).
There were no further indicators to decide between 1) and 2). Only by assuming that the help of the experimentators did not influence the outcome, some of the conclusions concerning the occurrence of problems in the previous sections can be drawn. However, when working with the conclusions, the here described threat needs to be kept in mind.

**Construct validity**

I asked the students to rank the design alternatives after initial reading before conducting the modelling and analysis. Most students were not able to guess the best design alternative correctly. Therefore it can be argued that the decision for a specific design alternative was unclear beforehand. However, for the Media Store, the majority of participants identified either one of the two alternatives in the best cluster as being best. Still, the results were much better for the predictions (cf. results of metric [M1.3] in section 5.1.1).

For the rankings, the permutation score as defined in metric [M1.4] was also measured for the initially estimated rankings. Here, the score was again significantly higher than for the predictions. Still, the average score was better than expected for random rankings. Some participants estimated the correct ranking, less others the complete opposite. Thus, the participants were able to partly identify the correct ranking, but there were still misestimations.

Thus, for the used design alternatives, an educated guess partly allowed to draw right conclusions, but the systematic approach resulted in a much better evaluation of the alternatives. This result fit the requirements for the construct validity in terms of complexity of the design decisions.

**External validity**

As there were differences as well as similarities between the system (e.g., as visible in figure 5.5, overall duration vs. time to model the control flow), that were observed for both approaches and both experiment groups, the conclusions drawn here can probably be generalised to the class of similar systems. However, as a main similarity between the systems was their low complexity compared to real applications, it is still unknown whether the findings can be generalised to larger and more complex systems.

Another identified threat to validity was that participants had problems with the task description that changed their way of solving the tasks. Although questions were asked concerning the task, there is no evidence that the participants had particular difficulties. Most questions were asked to clarify e.g. the control flow.

However, as the participants had certain problems with the methodology, the results are not generalisable to experts with the tools, e.g. the inventors, who know every little detail of the methodology and how to solve or work around problems. However, as actual users of the methodologies are probably better represented by the participants than by experts, the results may still be generalisable to these persons. Results may even be better generalisable to these persons than findings in case studies found in publications (cf. section 3.1.2), because they were often conducted by the authors and inventors of the particular approach.
6 Conclusions and Outlook

In this chapter, I first summarise this thesis in section 6.1. Then, in section 6.2, I present the benefits and the lessons learned of this thesis. Finally, in section 6.3 I highlight open issues and starting points for future work.

6.1 Summary

This thesis empirically validates the applicability of the performance prediction approach Palladio. As a reference for the applicability, I compared it to the well-known SPE approach, which already has been applied in industry settings. I conducted the empirical study in form of a controlled experiment, in which 19 computer science students took part. They predicted the performance of two artificial component-based software systems each with five design alternatives, and ranked the design alternatives based on the predicted response time. The metrics measured the similarity of the created models to a reference model, assessed by comparing the predicted response times, and the duration of making a prediction and of the preparation.

The results showed that with both approaches, performance predictions can be conducted by the participants.

For the quality in terms of similarity to the reference model, it was found that the deviation of the predicted response time to the response time predicted with the reference model was in average about 8%, with slightly less deviation for Palladio. However, higher average deviations of up to 38% and 20% were measured for single variants for SPE and Palladio, respectively. Overall, the deviation differed significantly for different variants. Altogether, both approaches allowed the identification of the best design option and the ranking of design options, in particular much better than by estimating the ranking with an educated guess.

Different reasons were detected for the deviations from an optimal model. Problems related to the task, the tools and the methodology were differentiated. For Palladio, more problems occurred with the tool, whereas for SPE, more problems concerned the methodology. Additionally, for Palladio, problems occurred during creating the models and resulted in asked questions, whereas for SPE, problems were detected later in the acceptance test or in the final models.

The interpretation of the results was largely unproblematic for both approaches. Instead, the most problematic factor was the parametrisation for Palladio, as shown by more problems measured by the metrics and the lowest grade for comprehensibility as assessed by the participants. The most occurred problems for SPE was the overhead matrix. The modelling of distributed systems, however, received the lowest grade for SPE.
Both tools were largely unproblematic, although for both, a number of disadvantages were named that their developers may want to address in the future. The Palladio tool was preferred by the participants, although the participants had more problems with it than with SPE-ED.

Looking at the duration for making a prediction, participants using SPE were overall faster, and with Palladio, more participants were not able to finish the tasks. Interestingly, the durations for single variants differed across the approaches: Variants that were very time-consuming for Palladio were quickly solved in SPE and vice versa. The main effort for both approaches was the modelling, but for Palladio also a lot of time was needed for analysis (as simulation was used) and searching for errors.

Quality deviation and required effort were strongly influences by the variant under study, and differently for the two approaches. Both approaches had advantages and disadvantages for different variants. As the systems contained different variants each, this also lead to differences in quality deviation and required effort of the averaged results for the systems. Furthermore, the results for analysing the systems were different in the types of problems that occurred, which I traced back to the different nature, e.g. the different complexity of the control flow.

6.2 Knowledge gained

With this thesis, the applicability of Palladio was successfully empirically validated. Further insights into the factors that influence a successful application of the approaches have been gained: In this study, the nature of the systems under study and even the different design alternatives greatly influenced the outcome. Developers of the two approaches can draw valuable conclusions from this thesis and improve their approaches.

Additionally, the results help to understand the relation between the system under study and the applicability in general. SPE is advantageous to model straightforward control flows without too many parametric dependencies, and allows very fast predictions here. However, predictions become more laborious if complex parametrisation and distribution of components are added. Palladio, on the other hand, has a high initial effort to create even simpler models, as models contain more information. However, the specialised constructs such as parameters, passive resources and completions are advantageous if needed.

Beyond that, this thesis presented a design for an empirical study that compares two performance prediction approaches on their applicability. This design can be also used to assess and compare other performance prediction approaches. If conducted with the same experimental setting, the results may even be comparable to the results of this thesis. Thus, with the experiment design, a more general contribution to software engineering, more specific to the empirical validation of performance prediction approaches, has been achieved.
6.3 Future Work

"Good scientific work poses more questions than it provides answers.”
(Prof. Dr. Roland Vollmar)

Several points of contact for future work and open issues are posed by this thesis.

Firstly, in this thesis, the predictions by the participants were compared to a predictions conducted for a reference model to isolatedly assess the applicability of the approaches. An interesting enhancement would be to further implement the studied systems and conduct a type I validation for both Palladio and SPE. This allows (1) to assess which of the two approaches was actually right with its predictions for Web Server and Media Store, and (2) to assess how strong the deviations within the experiment groups were by assessing the deviation to the real measurements. The deviations measured in this thesis are more serious if the reference model very accurately reflects the measurements than if the reference model is deviating from the measurements anyway.

Secondly, the relation between the actual system and the actual design alternative under study, i.e. the actual scenario for a performance prediction, and the quality, duration and problems of a prediction can be further investigated. Here, it would be interesting to identify the characteristics of scenarios in which the two approaches are advantageous, and thus be able to generate guidelines telling software architects when to apply which approach.

Finally, the reusability of the models could be further assessed, especially if reused by other people. In this thesis, the maintainability of the models was evaluated subjectively by the participants, which lead to contradictory results. Some participants stated Palladio is better maintainable, others stated the opposite (cf. section 5.1.2). Here, further experimentation (possibly using the models created in this study) could lead to a better understanding of the reusability of the models for both approaches. Additionally, consideration between effort and reuse can be made.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Structure and line of reasoning of this thesis</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Generic process of performance analysis (cf. [RH06, p.312])</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Queueing networks (from [SW02, p.136 and p.141])</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Influences on component performance (derived from [RBK07, p.23])</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>Palladio process (from [BKR07b])</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Specification in the PCM (left) and graphical representation (right) of a random double variable</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>RDSEFF editor and response time histogram in the PCM Bench</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>The SPE Workflow (from [SW02, p.409])</td>
<td>22</td>
</tr>
<tr>
<td>2.8</td>
<td>Analysis views of the SPE-ED tool</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Experiment Principles (from Wohlin, [WRH00, p.32])</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Metric M2.1: Problem dimensions</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Self-assessed programming skills and number of semesters of the participants</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Programming experience and experience with performance analysis of the par-ticipants</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>Visited courses</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>Achieved points in preparatory exercises</td>
<td>59</td>
</tr>
<tr>
<td>4.5</td>
<td>Experiment design</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>Initial Media Store system</td>
<td>62</td>
</tr>
<tr>
<td>4.7</td>
<td>Predicted mean response time of the reference for the Media Store system using SPE-ED</td>
<td>66</td>
</tr>
<tr>
<td>4.8</td>
<td>Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 1 using Palladio</td>
<td>66</td>
</tr>
<tr>
<td>4.9</td>
<td>Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 2 using Palladio</td>
<td>67</td>
</tr>
<tr>
<td>4.10</td>
<td>Initial Web Server system</td>
<td>68</td>
</tr>
<tr>
<td>4.11</td>
<td>Two ways of modelling the Web Server usage profile</td>
<td>71</td>
</tr>
<tr>
<td>4.12</td>
<td>Predicted mean response time of the reference for the Web Server system us-ing SPE-ED</td>
<td>72</td>
</tr>
<tr>
<td>4.13</td>
<td>Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 1 using Palladio</td>
<td>73</td>
</tr>
<tr>
<td>4.14</td>
<td>Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 2 using Palladio</td>
<td>73</td>
</tr>
<tr>
<td>4.15</td>
<td>Experiment Principles (from Wohlin, [WRH00, p.64])</td>
<td>76</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.1</td>
<td>Metric M3.1: Duration of whole task for both approaches and both systems</td>
<td>102</td>
</tr>
<tr>
<td>5.2</td>
<td>Metric M3.1: Duration of the initial system only, for both approaches and both systems</td>
<td>102</td>
</tr>
<tr>
<td>5.3</td>
<td>Metric M4.1: Duration of the prediction for the different variants of Media Store</td>
<td>105</td>
</tr>
<tr>
<td>5.4</td>
<td>Metric M4.1: Duration of the prediction for the different variants of Web Server</td>
<td>105</td>
</tr>
<tr>
<td>5.5</td>
<td>Metric M4.2: Break down of the duration for the initial system</td>
<td>107</td>
</tr>
<tr>
<td>5.6</td>
<td>Example distributions with K-S test p-value of $4 \cdot 10^{-5}$ and 0</td>
<td>114</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Requirements for an approach to be comparable to Palladio</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Evaluation of component-based approaches on requirements for comparability</td>
<td>19</td>
</tr>
<tr>
<td>2.6</td>
<td>Evaluation of monolithic approaches on requirements for comparability</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Research Goal</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Summary GQM Questions and Metrics</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Threats to internal validity</td>
<td>78</td>
</tr>
<tr>
<td>4.4</td>
<td>Threats to external validity</td>
<td>80</td>
</tr>
<tr>
<td>5.1</td>
<td>Metric M1.1: Relative deviation of the predicted response times for Palladio</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>Metric M1.1: Relative deviation of the predicted response times for SPE</td>
<td>84</td>
</tr>
<tr>
<td>5.3</td>
<td>Metric M1.2: Passed K-S Test ratio of predicted response time distribution and reference: $\text{PassRatio}_{v,u,Pal}$</td>
<td>85</td>
</tr>
<tr>
<td>5.4</td>
<td>Metric M1.3: Average percentage of the correct design decisions of the initially estimated design options rankings</td>
<td>87</td>
</tr>
<tr>
<td>5.5</td>
<td>Clustering of Media Store results</td>
<td>88</td>
</tr>
<tr>
<td>5.6</td>
<td>Clustering of Web Server results</td>
<td>88</td>
</tr>
<tr>
<td>5.7</td>
<td>Metric M1.4 (redefined): Weighted average of the permutation score of the design options rankings</td>
<td>89</td>
</tr>
<tr>
<td>5.8</td>
<td>Metric M1.4: Average of the permutation score of the initially estimated design options rankings</td>
<td>89</td>
</tr>
<tr>
<td>5.9</td>
<td>Metric M2.1: Relative number of task related problems</td>
<td>91</td>
</tr>
<tr>
<td>5.10</td>
<td>Metric M2.1: Relative number of Palladio related problems</td>
<td>92</td>
</tr>
<tr>
<td>5.11</td>
<td>Metric M2.1: Relative number of SPE related problems</td>
<td>93</td>
</tr>
<tr>
<td>5.12</td>
<td>Metric M2.1: Relative number of problems separated by occurrence</td>
<td>94</td>
</tr>
<tr>
<td>5.13</td>
<td>M2.2: Average number of rejection before acceptance level is reached</td>
<td>94</td>
</tr>
<tr>
<td>5.14</td>
<td>M2.3: Average score for errors in interpreting results</td>
<td>95</td>
</tr>
<tr>
<td>5.15</td>
<td>M2.4: Subjective evaluation of the comprehensibility of the Palladio concepts</td>
<td>96</td>
</tr>
<tr>
<td>5.16</td>
<td>M2.4: Subjective evaluation of the comprehensibility of the SPE concepts</td>
<td>96</td>
</tr>
<tr>
<td>5.17</td>
<td>Metric M2.7: Subjective advantages and disadvantages of the PCM Bench</td>
<td>99</td>
</tr>
<tr>
<td>5.18</td>
<td>Metric M2.7: Subjective advantages and disadvantages of the SPE-ED tool</td>
<td>99</td>
</tr>
<tr>
<td>5.19</td>
<td>Metric M3.1: Duration of making a prediction</td>
<td>101</td>
</tr>
<tr>
<td>5.20</td>
<td>Metric M4.2: Main time consuming activities for the Media Store</td>
<td>107</td>
</tr>
<tr>
<td>5.21</td>
<td>Metric M4.2: Main time consuming activities for the Web Server</td>
<td>108</td>
</tr>
<tr>
<td>5.22</td>
<td>Metric M4.3: Subjective advantages and disadvantages of the automated transformation named by the participants</td>
<td>108</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>A.1</td>
<td>Durations of solving preparatory exercises</td>
<td>CXXXVII</td>
</tr>
<tr>
<td>C.13</td>
<td>M2.1 Number of questions concerning the experiment task</td>
<td>CCLXXI</td>
</tr>
<tr>
<td>C.15</td>
<td>M2.1 Number of questions concerning Palladio</td>
<td>CCLXXII</td>
</tr>
<tr>
<td>C.17</td>
<td>M2.1 Number of questions concerning SPE</td>
<td>CCLXXII</td>
</tr>
<tr>
<td>C.18</td>
<td>M2.1 Number of problems in the acceptance test concerning the experiment task</td>
<td>CCLXXIII</td>
</tr>
<tr>
<td>C.20</td>
<td>M2.1 Number of problems in the acceptance test concerning Palladio</td>
<td>CCLXXIII</td>
</tr>
<tr>
<td>C.22</td>
<td>M2.1 Number of problems in the acceptance test concerning SPE</td>
<td>CCLXXIV</td>
</tr>
<tr>
<td>C.23</td>
<td>M2.1 Number of errors in the model concerning the experiment task</td>
<td>CCLXXIV</td>
</tr>
<tr>
<td>C.25</td>
<td>M2.1 Number of errors in the model concerning Palladio</td>
<td>CCLXXV</td>
</tr>
<tr>
<td>C.27</td>
<td>M2.1 Number of errors in the model concerning SPE</td>
<td>CCLXXV</td>
</tr>
</tbody>
</table>
Bibliography


[BJH+05] David A. Bacigalupo, Stephen A. Jarvis, Ligang He, Daniel P. Spooner, Donna N. Dillenberger, and Graham R. Nudd. An investigation into the application of dif-


[Pea00] Karl Pearson. On a criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can reasonably be supposed to have arisen from random sampling. Philosophical Magazine, 50:157–175, 1900.


Acknowledgements

First of all, I would like to thank Ralf Reussner, Steffen Becker and Heiko Koziolek for their great mentoring of this thesis. They gave valuable advice, supported me in teaching Palladio, provided slides for the lectures and spent a lot of time pre-reading the thesis. Additionally, I thank Klaus Krogmann for helping along during the experiment sessions.

Additionally, thanks to Walter Tichy and Lutz Prechelt for reviewing the experiment design and giving helpful comments, and Willi Hasselbring for examining this thesis in Oldenburg.

I also want to thank my parents, who supported me all my life in whatever I wanted to do, and were always a safe harbour to me.

Finally, thanks to all my friends, especially Jan Zeh, who had patience with me when I spent time on my studies.
Declaration

This thesis is my own work and contains no material that has been accepted for the award of any other degree or diploma in any university.

To the best of my knowledge and belief, this thesis contains no material previously published by any other person except where due acknowledgment has been made.

Oldenburg, 14th November 2007,

Anne Martens