CARL VON OSSIETZKY universität OLDENBURG

Department of Computing Science
Software Engineering Group

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

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 abovementioned 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?

[1] Note that I understand "quality" to be "similarity to a reference model" in this thesis, cf. section 3.2.2

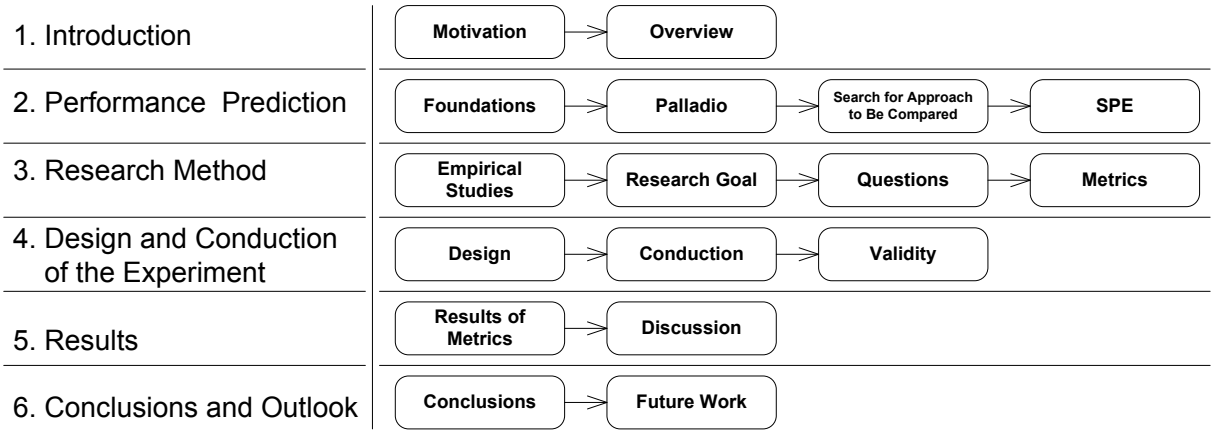


Figure 1.1: Structure and line of reasoning of this thesis

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 described 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 meets 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

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

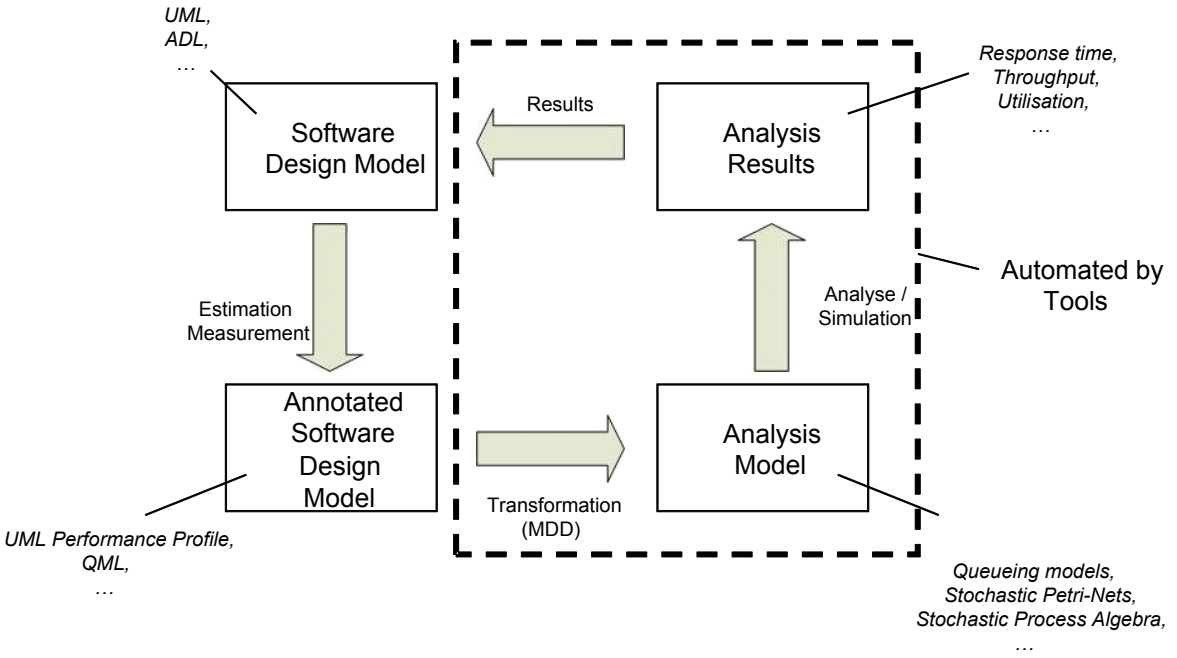


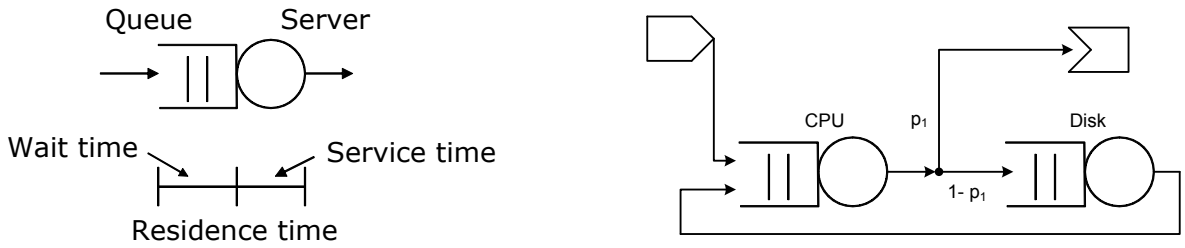
Figure 2.1: Generic process of performance analysis (cf. [RH06, p.312])

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



(a) Server and prefixed queue

(b) A simple queueing network

Figure 2.2: Queueing networks (from [SW02, p.136 and p.141])

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.

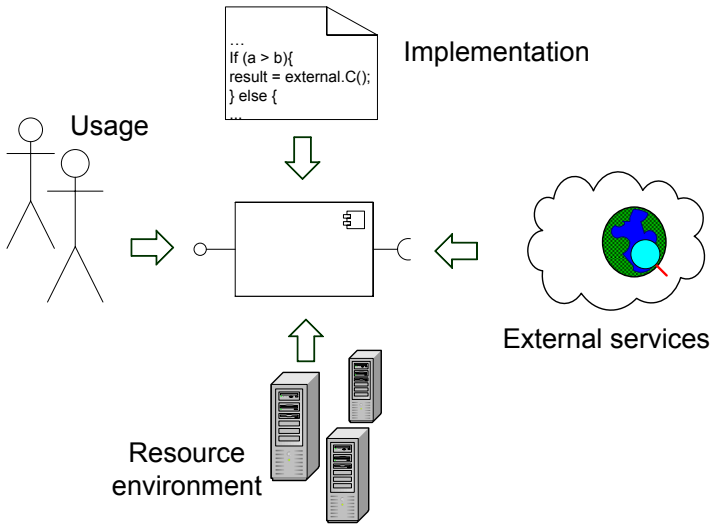


Figure 2.3: Influences on component performance (derived from [RBK+07, p.23])

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 [BKR07a] 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

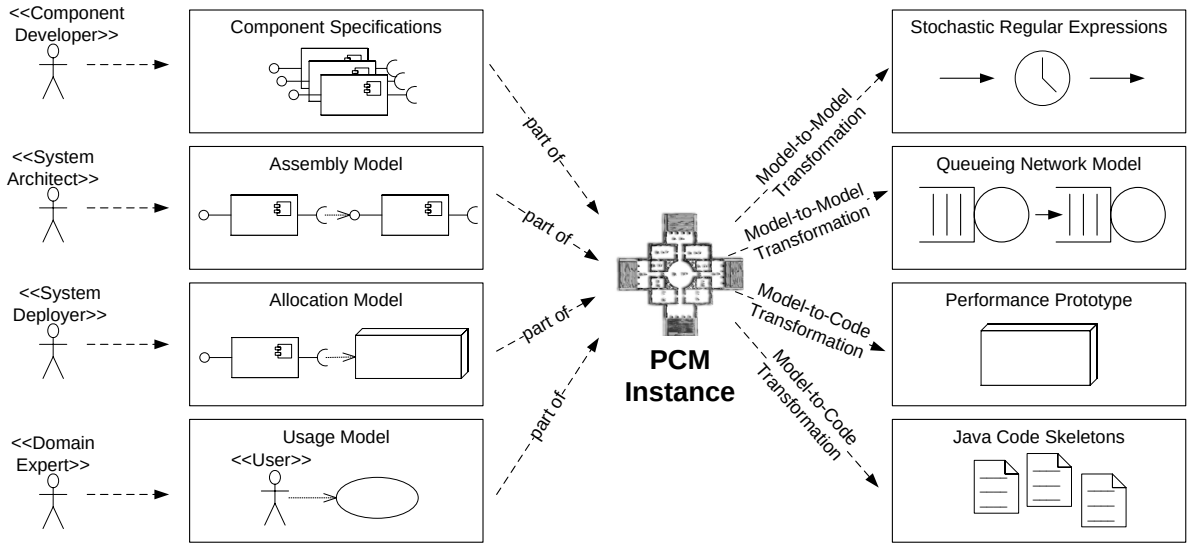


Figure 2.4: Palladio process (from [BKR07b])

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 [BKR07b].

Component developers specify interfaces, components, and datatypes. They create parametric models called Resource Demanding Service Effect Specification (RDSEFF) [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. section2.1). 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.

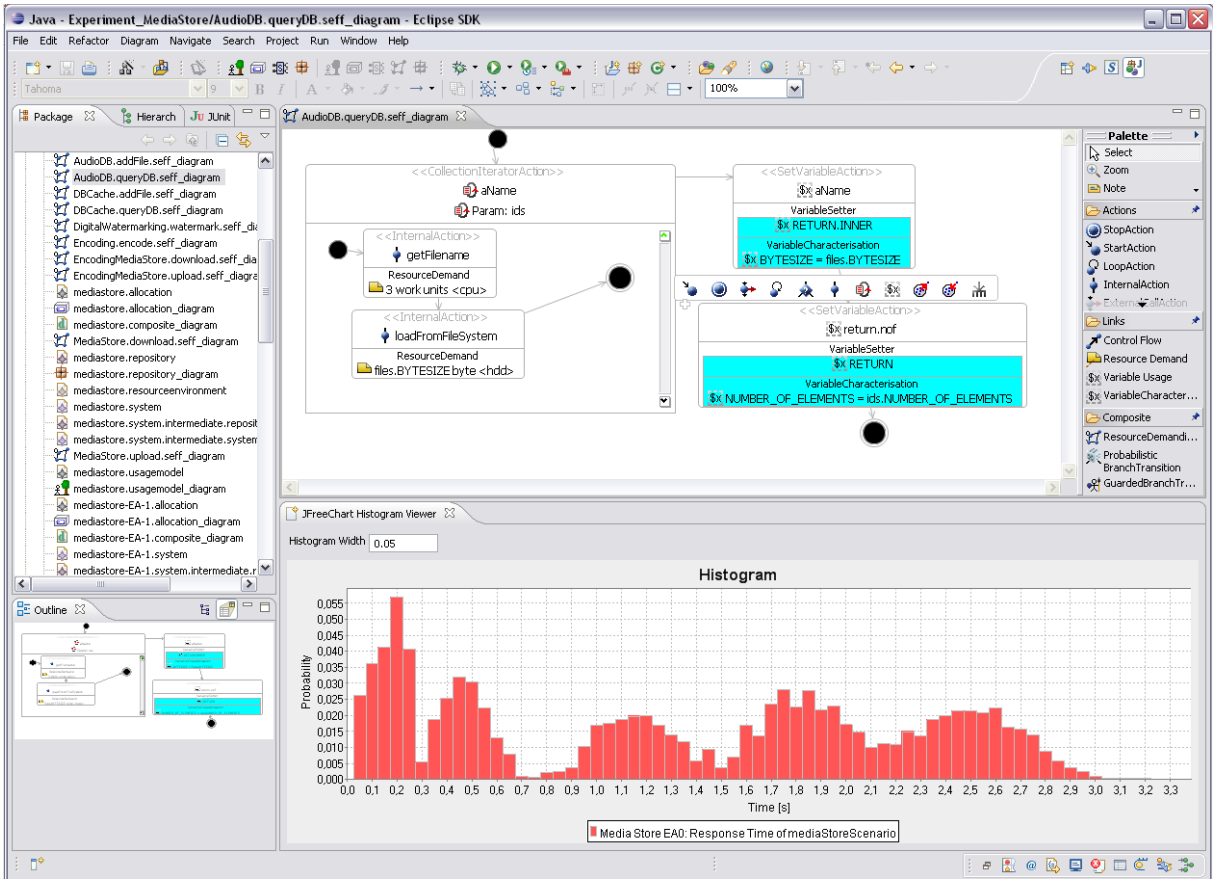


Figure 2.6: RDSEFF editor and response time histogram in the PCM Bench

Requirement	Rationale
Applicable at design time	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.
Usable tools	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.
Normal-case analysis	In contrast to worst-case or best-case analyses, the compared approach must allow to study the system under normal, average conditions.
Component-based	As Palladio is a component-based approach, it is desirable to compare it to another component based approach.
Support of arbitrary distributions	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.

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

Approach	Applicable at design time	Usable tools	Normal-case analysis	Component-based	Arbitrary distributions
KLAPER	✓	X	✓	✓	X
LQN-Components	✓	X	✓	✓	X
PACC	✓	X	✓	✓	X
ROBOCOP	✓	✓	X	✓	X
CBSPE	✓	X	✓	✓	X

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

experiment is questionable.

PACC [HMSW02] aims at predicting non-functional properties for embedded, safety- and time-critical component-based systems. Performance predictions are theoretically supported as well. However, the available model checking tool Component Formal Reasoning Technology (ComFoRT) only supports checking safety, reliability, and security requirements [IS04].

ROBOCOP provides a component model for embedded systems with real-time constraints [BdWC04]. It allows performance predictions and even guarantees for timing behaviour of real-time applications. Important aspects of the approach are synchronization issues in embedded systems. Tool support exists with the Robocop Integration Environment (RIE) tool and the DeepCompass framework [BCdK07]. However, as the approach focusses embedded systems and thus conducts worst-case analyses, it is unsuited to be compared to Palladio in the business information systems domain, looking at normal cases.

CBSPE [BM04] is based on the SPE process, and adapts it for component-based software development. The main adaptation is that the roles of component developer and system assembler are distinguished, so that both can work independently and only exchange specifications of the components. All in all, the approach resembles Palladio quite well. However, in a previous experiment ([Mar05]), I experienced that the accompanying tool based on ArgoUML was not mature enough and caused many inexplicable problems when being applied. As tool development has been cancelled, I did not use CB-SPE in this thesis.

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

Approach	Applicable at design time	Usable tools	Normal-case analysis	Component-based	Arbitrary distributions
SPE	✓	✓	✓	X	X
PRIMA-UML	✓	X	✓	X	X
CSM	✓	?	✓	X	X
UML-PSI	✓	X	✓	X	X

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 approach 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

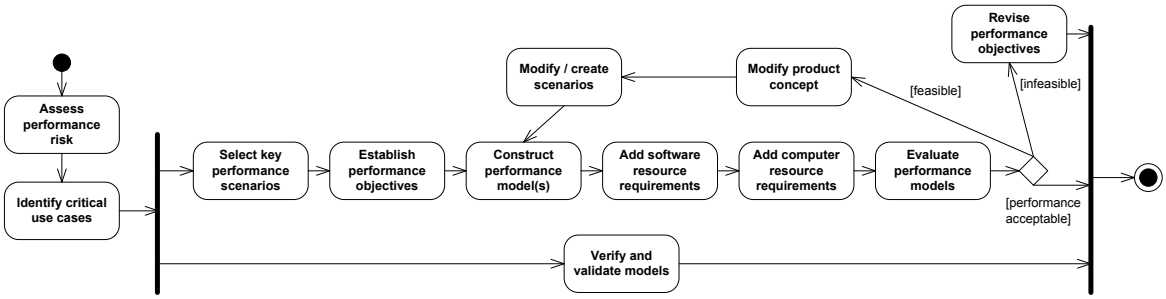


Figure 2.7: The SPE Workflow (from [SW02, p.409])

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, modifing 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.

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

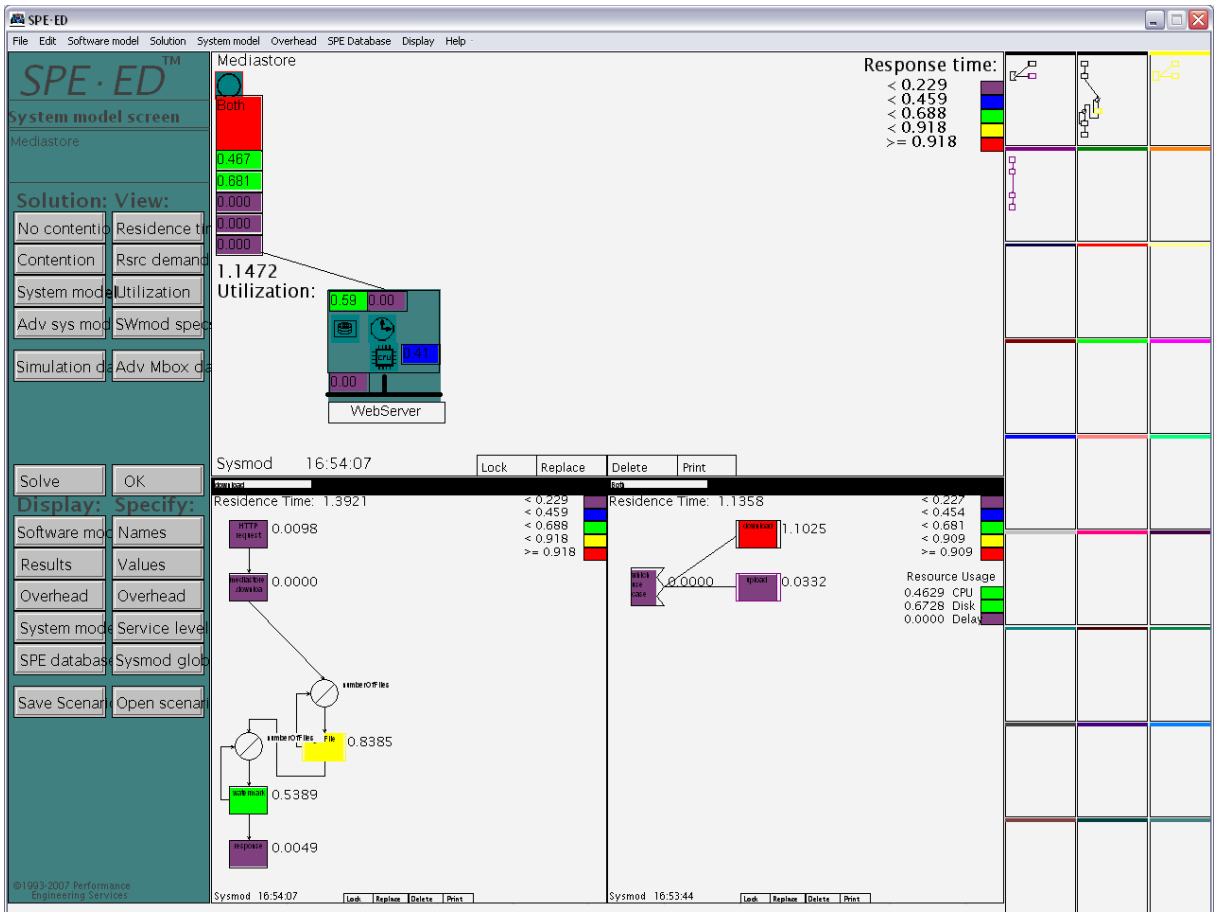


Figure 2.8: Analysis views of the SPE-ED tool

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

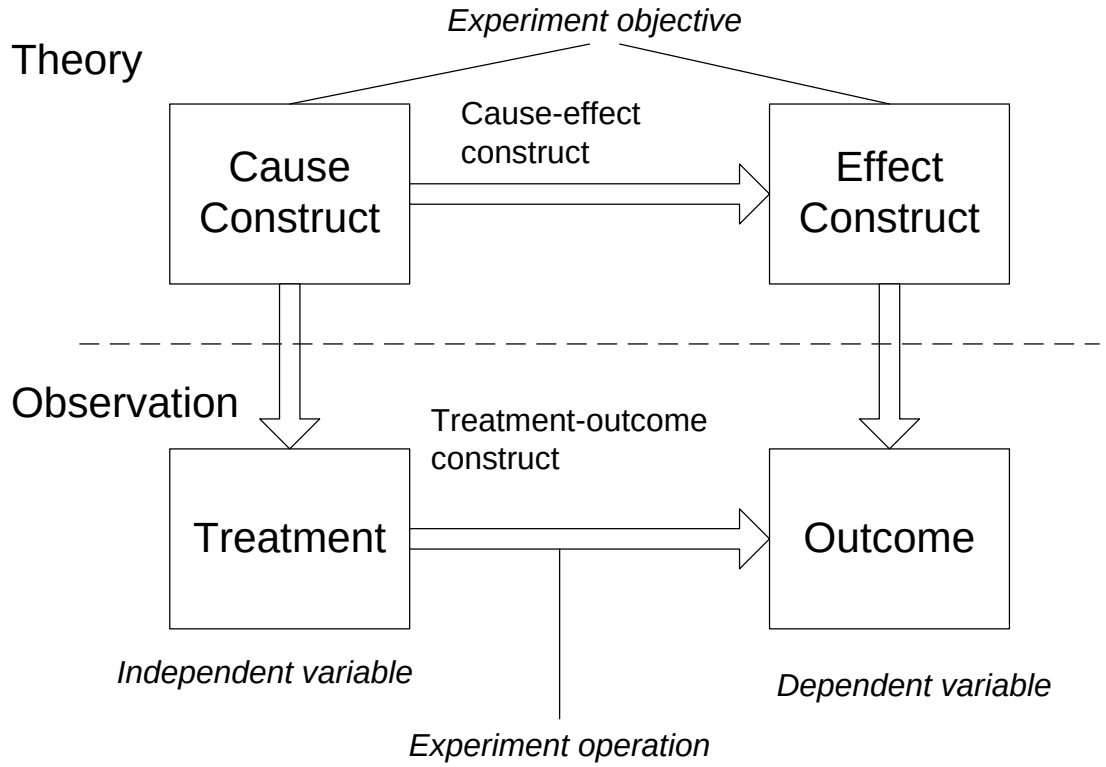


Figure 3.1: Experiment Principles (from Wohlin, [WRH+00, p.32])

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, BJH+05]. 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.

Purpose	Empirically validate
Issue	the applicability
Object	of the performance prediction approach Palladio
Viewpoint	from a user's point of view

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 $s \in S$: $V^s = \{v_1^s, v_2^s, v_3^s, v_4^s, v_5^s\}$
- 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.

Question 1	What is the quality of the created performance prediction models?
Metric M1.1	Relative deviation of predicted mean response times of the participants and of the reference model.
Metric M1.2	Passed K-S Test ratio of predicted response time distribution and reference
Metric M1.3	Percentage of correct design decisions.
Metric M1.4	Permutations in design decision rankings, recognising clusters.
Hypothesis 1	With both approaches, the created models are similar to the reference model.
Question 2	What are the reasons for the model's quality?
Metric M2.1	Problems when creating the models and classification
Hypothesis 2	Some potential problems arise from a lack of understanding and tool difficulty.
Question 2.1	Are the approaches comprehensible?
Metric M2.2	Number of times of rejection before acceptance level is reached.
Metric M2.3	Number of interpretation problems
Metric M2.4	Subjective evaluation of comprehensibility by the participants.
Metric M2.5	Subjective evaluation of distribution functions by participants.
Metric M2.6	Subjective evaluation of parametrisation by participants.
Question 2.2	Are the tools usable?
Metric M2.7	Subjective evaluation of the tool usability by participants.
Question 2.3	What are further reasons?
Metric M2.8	Analysis of explanations in questionnaire to find additional influences.
Question 3	What is the duration of predicting the performance?
Metric M3.1	Average duration of a prediction
Metric M3.2	Time needed to solve preparatory exercises
Metric M3.3	Subjective evaluation by participants on needed time and effort to learn the approaches
Hypothesis 3	The duration for a Palladio prediction is 1.5 times higher as the duration for an SPE prediction.
Question 4	What are the reasons for the duration?
Metric M4.1	Duration of the single steps
Metric M4.2	Breakdown of the duration to activities
Metric M4.3	Subjective evaluation by participants on reasons for the needed time
Hypothesis 4	The most time-consuming activity is the modelling.

Table 3.2: Summary GQM Questions and Metrics

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 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 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 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 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 M1.1: Relative deviation of predicted mean response times of the participants and of the reference model. For metric 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 $predMeanResp_{v,u,p}$ 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 $refMeanResp_{v,u,a}$ 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:

$$absDevMeanResp_{v,u,a} = avg(\{|predMeanResp_{v,u,p} - refMeanResp_{v,u,a}| \,|s \in S, v \in V^s, p \in P_{a,s}\})$$

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

$$propDevMeanResp_{v,u,a} = \frac{absDevMeanResp_{v,u,a}}{refMeanResp_{v,u,a}}$$

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:
$$propDevMeanResp_a = avg(\{propDevMeanResp_{v,u,a} \,|s \in S, v \in V^s, u \in UP\})$$

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 χ^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 χ^2 test was not used.

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:

$$\text{Metric M1.3: } perc_a = \frac{\left|\left\{d \,\middle|\, d \in \bigcup_{s \in S, u \in UP} DD_{s,u,a}, d \text{ is correct}\right\}\right|}{\left|\bigcup_{s \in S, u \in UP} DD_{s,u,a}\right|}$$

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

$$PosReference_{s,u,a} : V^s \rightarrow 1, ..., |V^s|$$

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

$$class_{s,u,a} : 1, ..., |V^s| \rightarrow 1, ..., |classes|$$

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

$$perm_{u,p}(variant) = |class(PosPred_{u,p}(variant)) - class(PosReference_{s,u,a}(variant))|$$

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

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

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 $class(r_{ref}(A4)) = 3$ and $class(r_1(A4)) = 1$, a difference of 2 = $perm(A4)$. Overall, r_1 has an permutation of $perm = 2 + 0 + 2 + 0 = 4$.

Thus, the permutations of a whole ranking can be given as the sum $perm_{s,u,p}$. 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 $perm_{s,u,i}$. Now, the proportion of permutations was given with:

$$propPerm_{s,u,p} = \frac{perm_{s,u,p}}{perm_{s,u,i}}$$

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

$$\text{Metric M1.4: } propPerm_a = avg(\{propPerm_{s,u,p} \,|s \in S, p \in P_{a,s}, u \in UP|\})$$

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

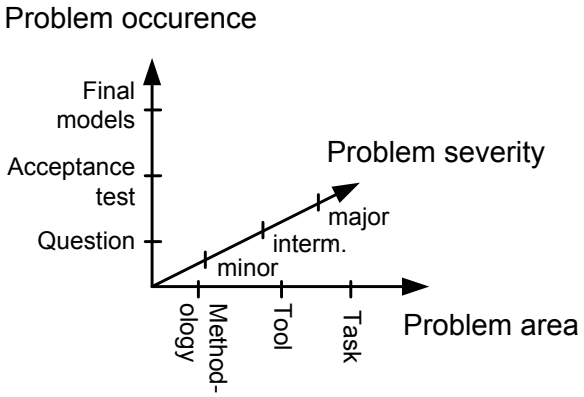


Figure 3.2: Metric M2.1: Problem dimensions

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

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 $Acc\#_p$ be the number of acceptance tests that

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

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

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 participants 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

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

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

$$\text{Metric M2.3: } propIntErr_a = avg(\{propPerm'_{s,u,p} \,|s \in S, p \in P_{a,s}, u \in UP|\})$$

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.

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 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 M3.1 measured the average duration over all participants.

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

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

Metric 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 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 $e \in E$.

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

Metric 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 the questionnaire in appendix B.4.3 (question 24) and formed metric 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 M4.1. Metric 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 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

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 $act \in Act = \{ra, cf, rd, re, up, err, ana\}$ for a variant of a system $v \in V^s, s \in S$. The duration $dact_{v,u,p,act}$ 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:

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

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

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 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:

$$\text{Metric M4.2: } dact_{a,act} = avg(\{dact_{v,u,a,act} \,|s \in S, v \in V^s|\})$$

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.

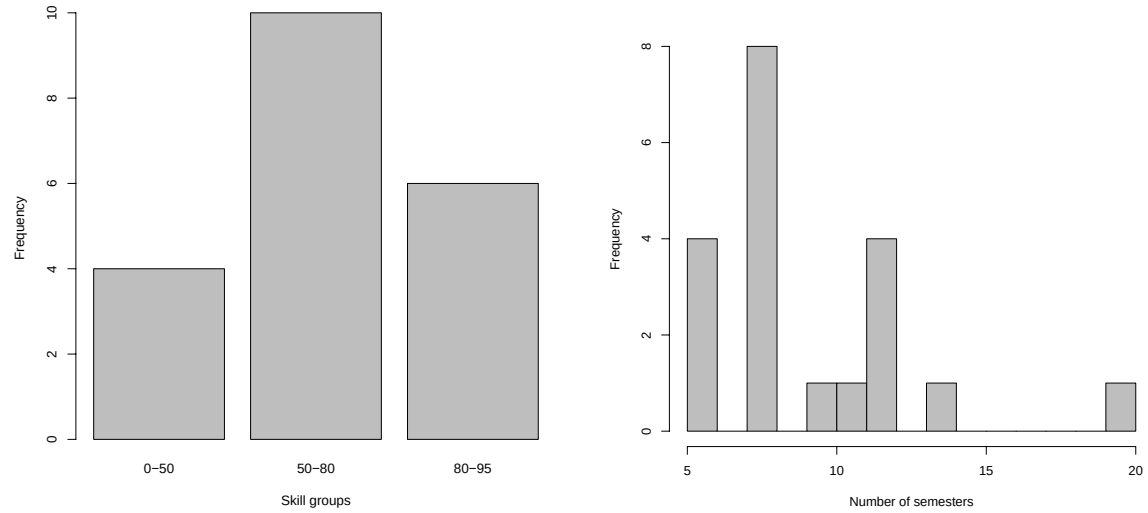
This chapter describes the experimental set-up, as well as the required preparations. In section 4.1, I describe the participants and their training. In section 4.2, I describe the actual experiment. Finally, I discuss the validity of the experiment in section 4.3.

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



(a) Self-assessed programming skills

(b) Number of semesters

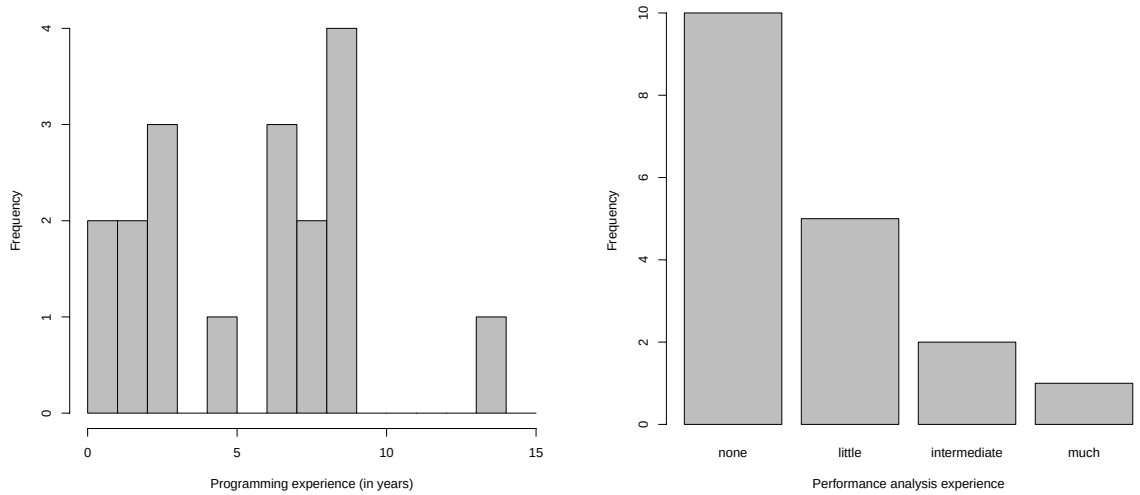
Figure 4.1: Self-assessed programming skills and number of semesters of the participants

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].



(a) Programming experience (b) Experience with performance analysis

Figure 4.2: Programming experience and experience with performance analysis of the participants

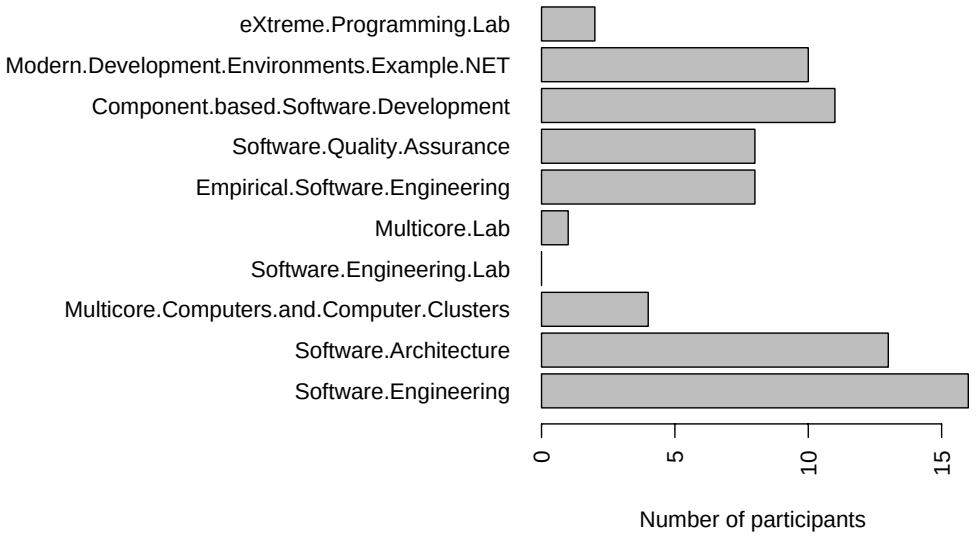


Figure 4.3: Visited courses

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.

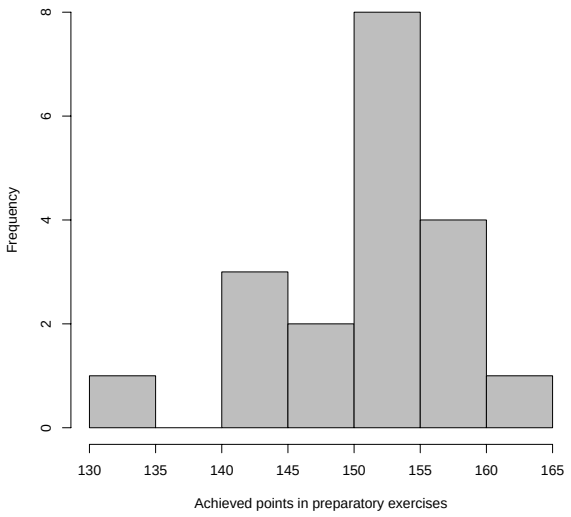


Figure 4.4: Achieved points in preparatory exercises

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 counter-balanced 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].

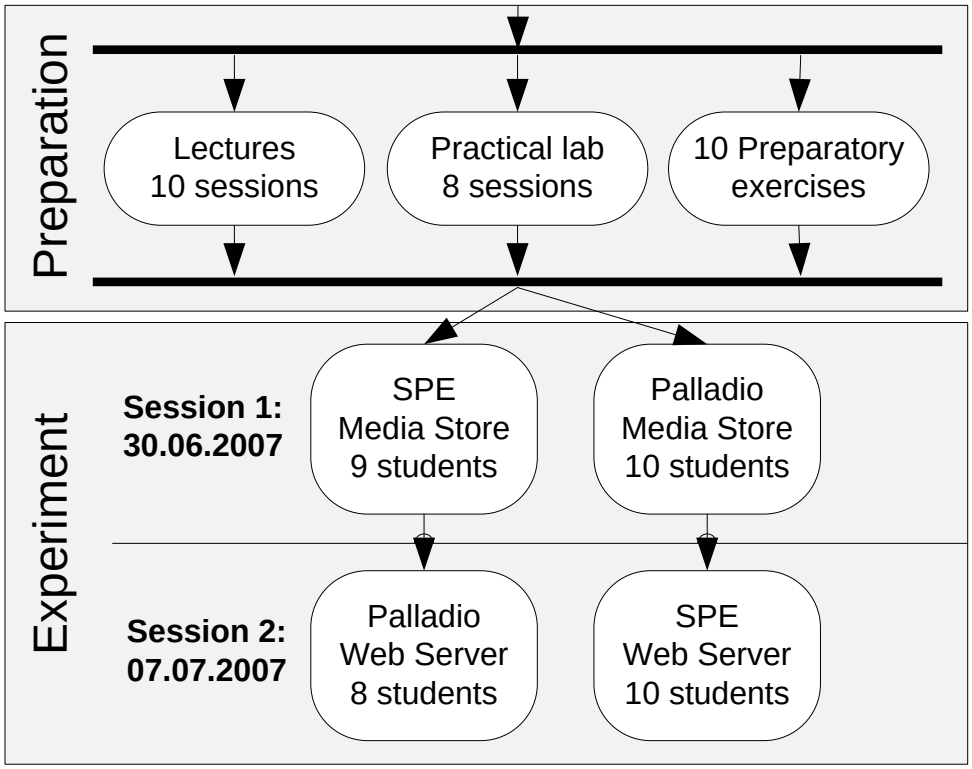


Figure 4.5: Experiment design

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 they 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].

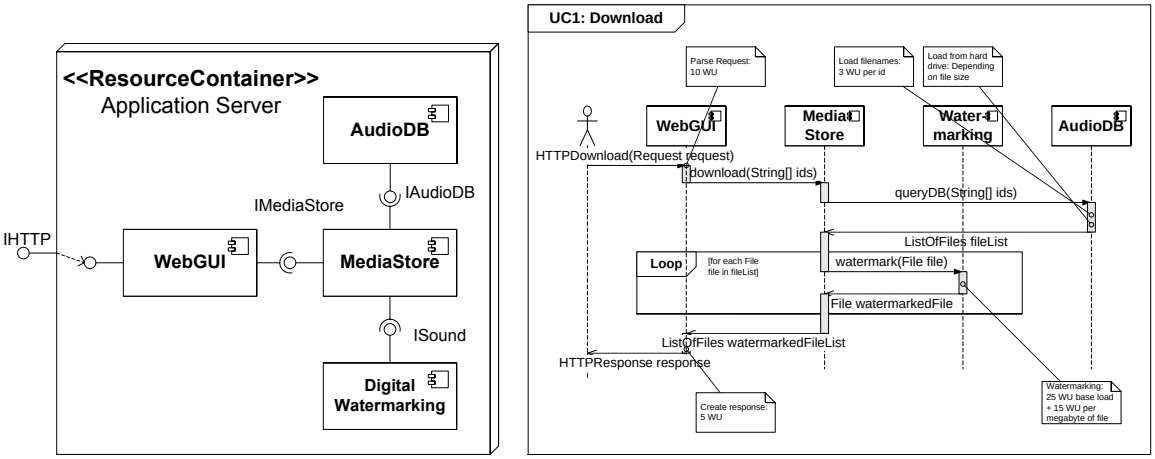
4.2.2 Experimental Tasks

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,



(a) Assembly of the Media Store components (b) Sequence diagram for downloading use case

Figure 4.6: Initial Media Store system

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_1^{MS}): 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_2^{MS}): 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_3^{MS}): 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 options allowed to analyse distributed systems and network communication.

Re-encoding and reduction of bit rate (v_4^{MS}): For this option, a new `Encoding` component was introduced and the `MediaStore` component was replaced by a `EncodingMediaStore` 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 (v_5^{MS}): 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.

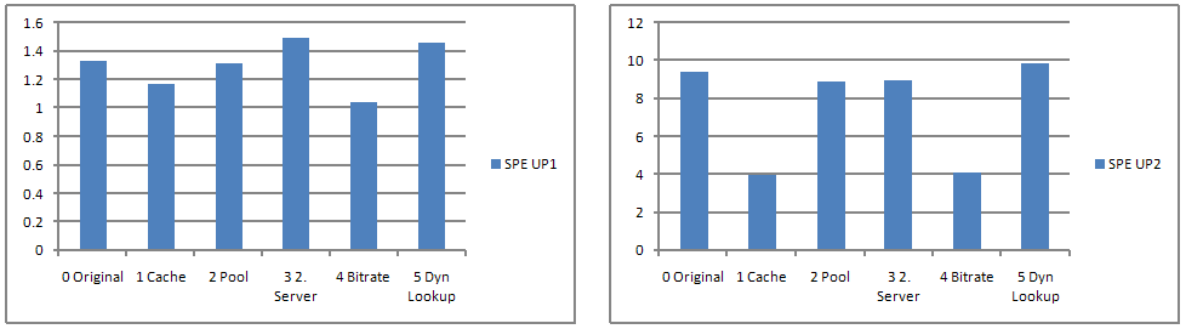
Reference Model I modelled the Media Store using SPE and Palladio, to be used as a reference for the acceptance test and for the comparison with the models created by the participants.

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 $refMeanResp_{v,u,SPE}$. 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 $refMeanResp_{v,u,Pal}$ 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 (v_4^{MS}). In the SPE-ED predictions, this option was deemed 12 ms faster that the second-best option, the cache (v_1^{MS}, 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 (v_4^{MS}) was only 1.4 ms lower than the mean for the cache option (v_1^{MS}).

For usage profile 2, the best two design options were likewise very similar in respect to predicted response times. Here, the cache option (v_1^{MS}) had a little faster response time: In the SPE-ED predictions, the difference to the bit rate conversion variant (v_4^{MS}) was 11 ms (see figure 4.7(b)). In the Palladio prediction, the difference was again smaller. Again, looking at the CDF, the best



(a) $UP1$ (b) $UP2$

Figure 4.7: Predicted mean response time of the reference for the Media Store system using SPE-ED

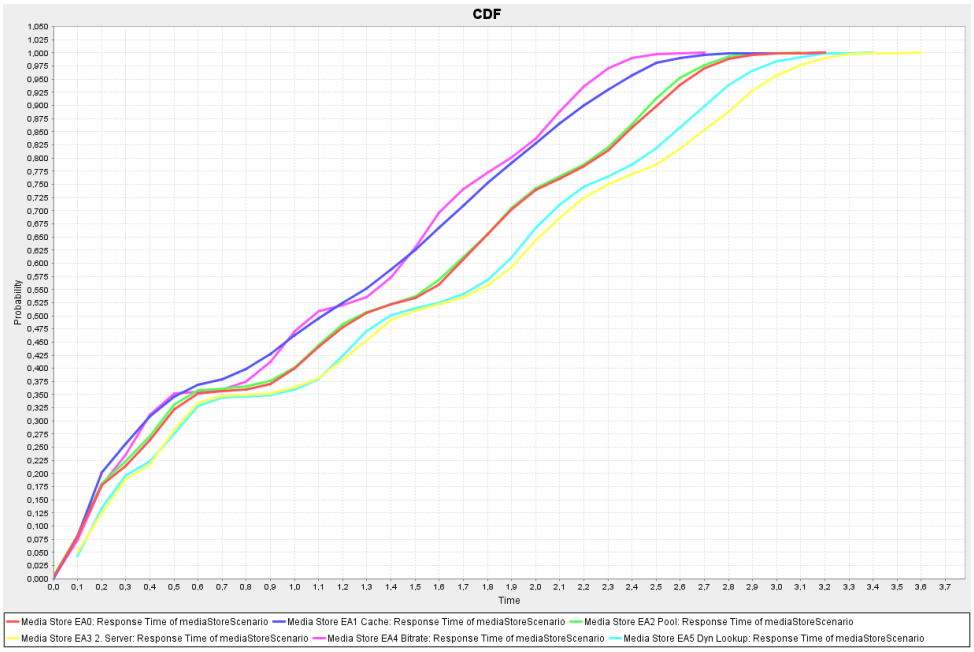


Figure 4.8: Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 1 using Palladio

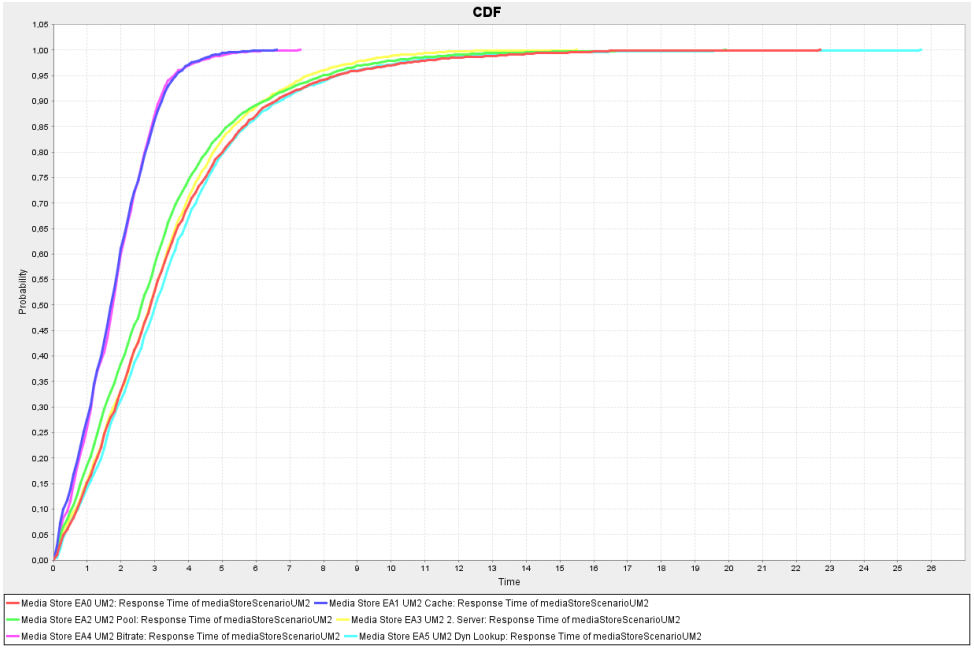


Figure 4.9: Predicted cumulated response time distribution of the reference for the Media Store system and usage profile 2 using Palladio

alternative cannot be read off (see figure 4.9). The mean of the distributions was lower for the cache option (v_1^{MS}) by 2.7 ms.

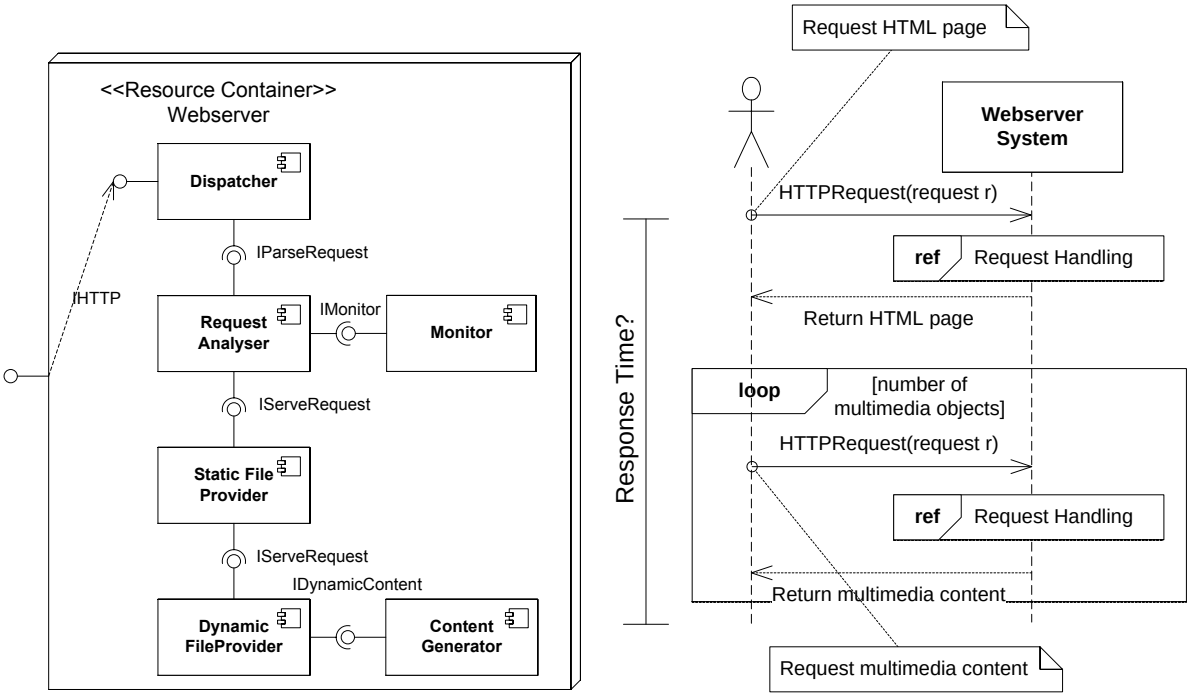
Web Server

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



(a) Assembly of the Web Server components

(b) User server interaction

Figure 4.10: Inital Web Server system

sizes 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 trade-off 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 a `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_5^{WS}): 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.

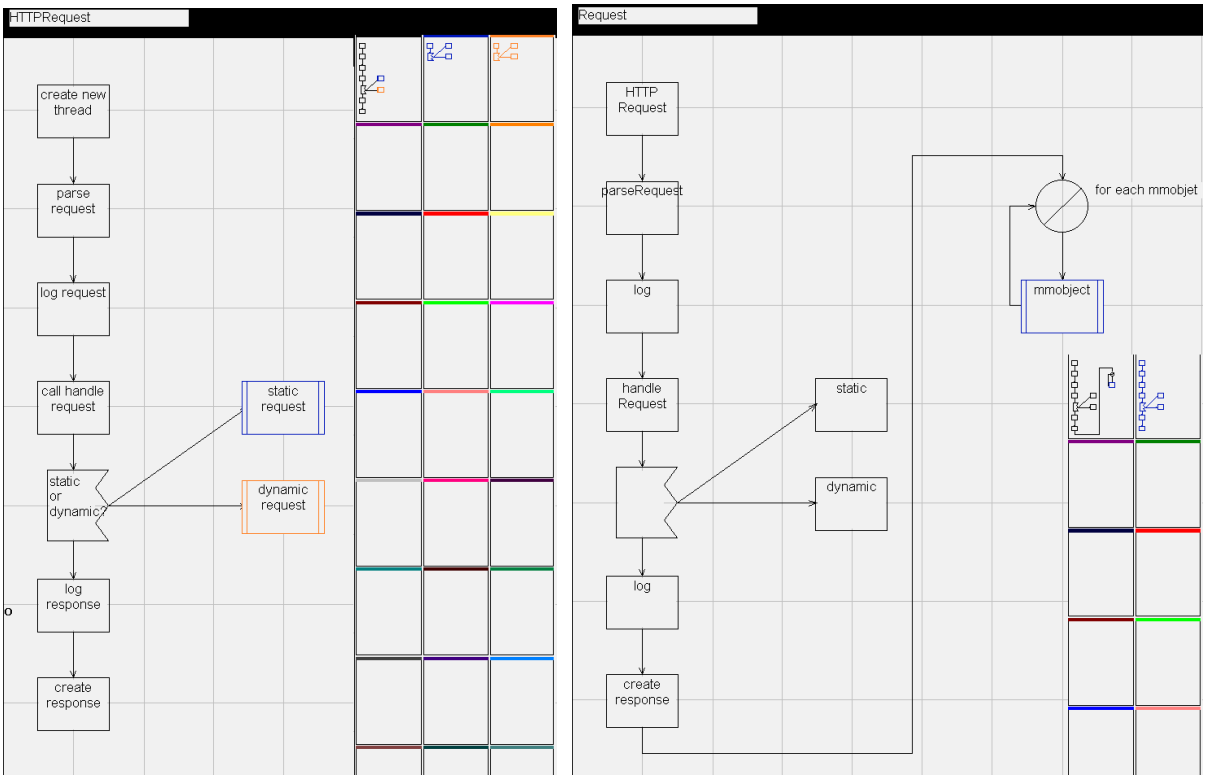
The use of this design option is again reflecting a common measure to improve performance, even if in a very simplified way. Usually, the question for such systems is not whether to use a thread pool, but rather to choose the optimal pool size [CGLL02]. Additionally, this option included the use of passive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

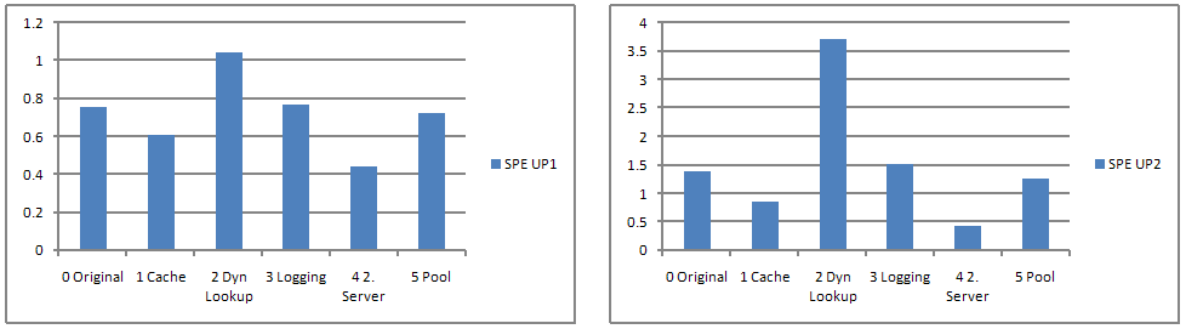
However, there was an alternative way of modelling. The usage profile could be unreeled to form a longer execution graph (as shown in figure 4.11(b)). Here, first the actions when an HTML page was requested were modelled, and then, in a loop, the same actions were repeated, only with different performance 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



(a) $UP1$ (b) $UP2$

Figure 4.12: Predicted mean response time of the reference for the Web Server system using SPE-ED

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_4^{WS}) 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_4^{WS}) mostly is above the CDF of the cache variant (v_1^{WS}).

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_4^{WS}) had the best predicted response time, the cache option (v_1^{WS}) was second-best. For the SPE approach, the predicted response time for the second server option (v_4^{WS}) 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_4^{WS}) always was above the CDF of the cache variant (v_1^{WS}, 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

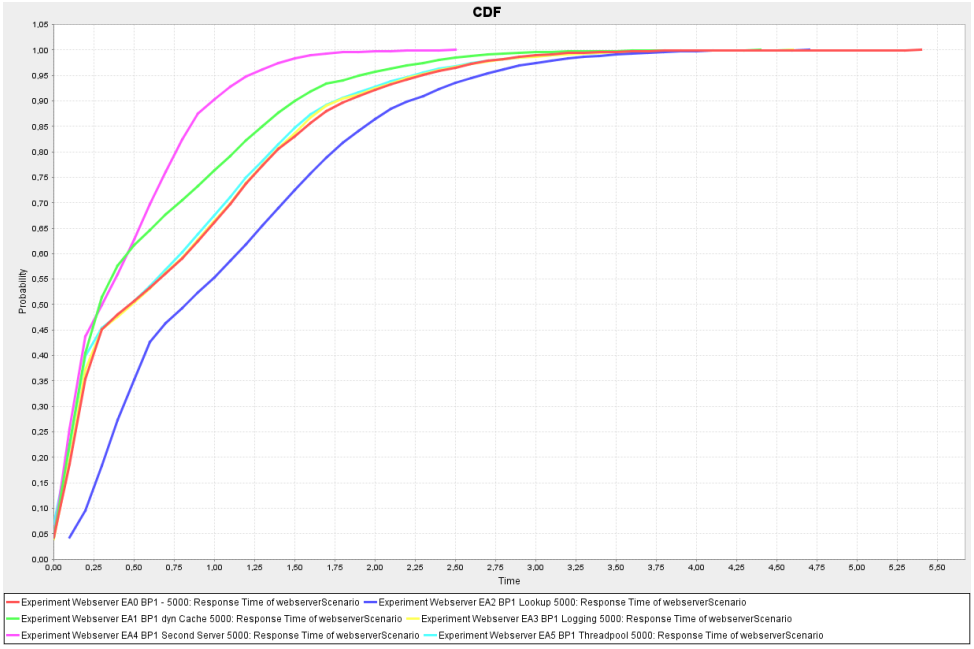


Figure 4.13: Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 1 using Palladio

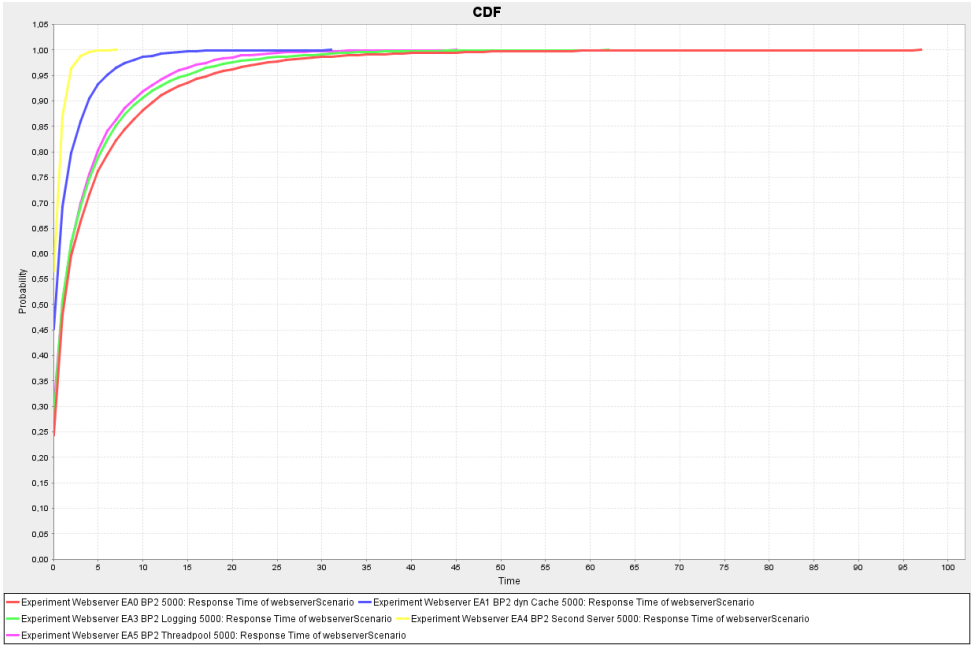


Figure 4.14: Predicted cumulated response time distribution of the reference for the Web Server system and usage profile 2 using Palladio

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 experimentator 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].

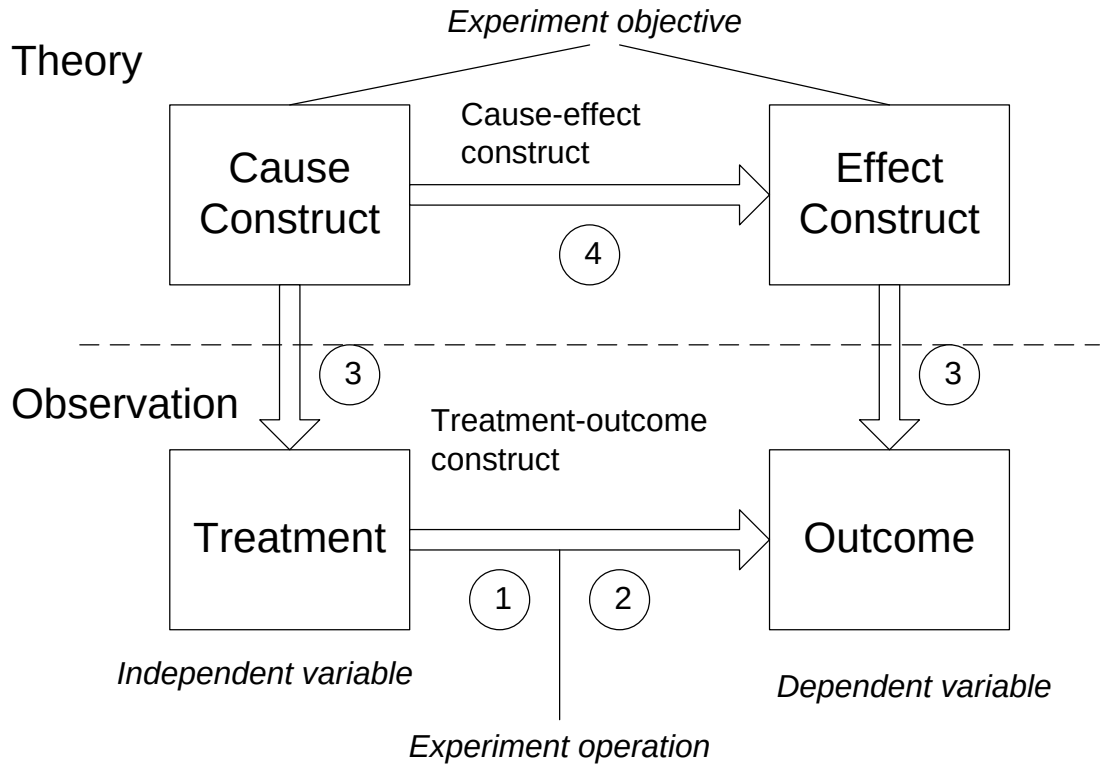


Figure 4.15: Experiment Principles (from Wohlin, [WRH+00, p.64])

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+00].

The conclusion validity was problematic in this thesis, as a relatively small number 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

Threat	Measures
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

Table 4.2: Threats to internal validity

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

Threat	Appraisal / Measures
Only small systems	• Inherent problem of experiments
Influence of the system under study	• Two different experimental tasks
Realistic way to solve tasks	• Achieve motivation, certain stress and feasible task
Problems with the task description	• Test task description beforehand

Table 4.4: Threats to external validity

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 be 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.

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

Table 5.1: Metric M1.1: Relative deviation of the predicted response times for Palladio

		v_0^s	v_1^s	v_2^s	v_3^s	v_4^s	v_5^s	Avg
Media Store	$UP1$	8.31%	9.58%	13.18%	11.59%	15.49%	9.95%	11.35%
	$UP2$	4.10%	9.49%	5.74%	21.22%	12.54%	8.17%	10.21%
Web Server	$UP1$	0.34%	1.28%	2.83%	2.15%	6.33%	1.56%	2.42%
	$UP2$	1.01%	1.22%	8.29%	4.47%	37.92%	2.33%	9.21%
Overall $propDevMeanResp_{SPE}$								8.3%

Table 5.2: Metric M1.1: Relative deviation of the predicted response times for SPE

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^s (second server), had a very high deviation, and v_0^s for the $UP2$, too. For the Web Server and Palladio, the deviations for the v_2^s, the broker alternative, was very high, and for $UP2$, additionally the deviations for v_0^s, v_1^s (Cache) and v_3^s (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^s (Reduction of the bit rate) and $UP1$, and exceptionally high compared to the other variants for v_3^s (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^s (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 $PassRatio_{v,u,Pal}$, for the different variants, usage profiles and systems. The overall average $PassRatio_{Pal}$ 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

		v_0^s	v_1^s	v_2^s	v_3^s	v_4^s	v_5^s	Avg
Media Store	$UP1$	0.71	0.29	1.00	0.00	0.33	0.71	0.55
	$UP2$	0.00	0.29	0.60	0.25	0.00	0.86	0.36
Web Server	$UP1$	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$UP2$	0.00	0.00	0.13	0.29	0.00	0.67	0.16
Overall average $PassRatio_{Pal}$					0.24			

Table 5.3: Metric M1.2: Passed K-S Test ratio of predicted response time distribution and reference : $PassRatio_{v,u,Pal}$

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 $perc_{s,u,a}$.

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 participants 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 $perc_{MS,u,a}$ 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 $perc_{cl,s,u,a}$ that measured the correct design decisions recognising the clusters. This evaluated to:

$$perc_{cl,MS,u,a} = 1, u \in UP, a \in A$$

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: $perc_{WS,UP1,SPE} = 1$.
- 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 $perc_{WS,UP1,Pal} = 0.67$.

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 $perc_{WS,UP2,SPE} = 0.88$.
- For Palladio, 5 out of 5 participants who ranked all variants did so: $perc_{WS,UP2,Pal} = 1$.

Combined I evaluated $perc_a, a \in A$ for the original definition of the metric M1.3 and as $perc_{cl,a}, a \in A$ 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.

$$\begin{aligned} perc_{SPE} &= 0.8 \\ perc_{Pal} &= 0.77 \\ perc_{cl,SPE} &= 0.97 \\ perc_{cl,Pal} &= 0.85 \end{aligned}$$

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 $percInitial_s$ and $percInitial_{cl,s}, s \in S$, that are defined analogously to $perc_a$ and $perc_{cl,a}$. 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 $percInitial_s$

$percInitial_s$	Media Store	Web Server
$UP1$	0.33	0.35
$UP2$	0.28	0.35
Average	0.31	0.35

(b) With recognising clusters $percInitial_{cl,s}$

$percInitial_{cl,s}$	Media Store	Web Server
$UP1$	0.78	0.35
$UP2$	0.61	0.35
Average	0.69	0.35

Table 5.4: Metric 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 $propPerm_{s,u,p}$ by the number of ranks assessed in the average calculation. Thus, complete rankings had a higher influence on the overall metric $propPerm_a$, 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 $V_p^s \subseteq V^s$ be the set of variants participant $p \in P$ defined a ranking for. I redefined metric M1.4 as given below. This redefinition did not change the outcome if all participants assigned all ranks in their ranking.

$$\text{Metric M1.4: } propPerm_a = \frac{\sum_{s \in S, p \in P_{a,s}, u \in UP} propPerm_{s,u,p} \cdot \left|V_p^s\right|}{\sum_{s \in S, p \in P_{a,s}, u \in UP} \left|V_p^s\right|}$$

	$u = UP1$, $a = SPE$	$u = UP2$, $a = SPE$	$u = UP1$, $a = Pal$	$u = UP2$, $a = Pal$
$class_{MS,u,a}(v_1^{MS})$, i.e. Cache	1	1	1	1
$class_{MS,u,a}(v_2^{MS})$, i.e. Pool	2	2	2	2
$class_{MS,u,a}(v_3^{MS})$, i.e. 2nd server	3	2	3	2
$class_{MS,u,a}(v_4^{MS})$, i.e. Bit rate	1	1	1	1

Table 5.5: Clustering of Media Store results

	$u = UP1$, $a = SPE$	$u = UP2$, $a = SPE$	$u = UP1$, $a = Pal$	$u = UP2$, $a = Pal$
$class_{WS,u,a}(v_1^{WS})$, i.e. Cache	2	2	2	2
$class_{WS,u,a}(v_3^{WS})$, i.e. Logging	3	4	3	4
$class_{WS,u,a}(v_4^{WS})$, i.e. 2nd server	1	1	1	1
$class_{WS,u,a}(v_5^{WS})$, i.e. Pool	3	3	3	3

Table 5.6: Clustering of Web Server results

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_1^{MS} to v_4^{MS}. 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_1^{WS} and v_3^{WS} to v_5^{WS} 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_4^{MS} and the cache component v_1^{MS}, 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_2^{MS} and second server v_3^{MS} 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_5^{MS} and the logging v_3^{MS} 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

$propPerm$	Media Store		Web Server		Weighted average
	UP1	UP2	UP1	UP2	$propPerm_a$
Palladio	0	0	0.11	0.09	0.065
SPE	0.04	0.1	0.08	0.06	0.073

Table 5.7: Metric M1.4 (redefined): Weighted average of the permutation score of the design options rankings

For estimated ranking	Media Store	Web Server	Average
$UP1$	0.32	0.33	0.33
$UP2$	0.25	0.34	0.30
Average	0.29	0.34	0.31

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

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

For the estimated initial rankings, the permutations score could be calculated as originally defined, as all participants ranked all alternatives. As mentioned above, for this case, the original and the redefined metric led to the same results. Table 5.8 shows the averaged permutation score for the initial estimation of the rankings.

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

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

Table 5.9: Metric M2.1: Relative number of task related problems

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-

	Tool				Methodology						
	Usage	Error	Bug	Sum	Parameters	Component parameters	Types and units	Assembly	Usage model	Sum	Sum
Media Store											
minor	0.00	0.00	0.00	0.00	0.00	0.43	0.14	0.00	0.00	0.57	0.57
intermediate	0.43	0.43	0.14	1.00	1.00	0.29	0.57	0.00	0.00	1.86	2.86
major	0.14	0.00	0.14	0.29	0.57	0.29	0.00	0.00	0.00	0.86	1.14
Sum	0.57	0.43	0.29	1.29	1.57	1.00	0.71	0.00	0.00	3.29	4.57
Web Server											
minor	0.25	0.00	0.25	0.50	0.00	0.25	0.00	0.00	0.00	0.25	0.75
intermediate	0.88	0.38	0.13	1.38	0.63	0.13	0.38	0.00	0.00	1.13	2.50
major	1.13	0.00	0.25	1.38	0.00	0.00	0.25	0.13	0.13	0.50	1.88
Sum	2.25	0.38	0.63	3.25	0.63	0.38	0.63	0.13	0.13	1.88	5.13
Both systems											
minor	0.13	0.00	0.13	0.25	0.00	0.34	0.07	0.00	0.00	0.41	0.66
intermediate	0.65	0.40	0.13	1.19	0.81	0.21	0.47	0.00	0.00	1.49	2.68
major	0.63	0.00	0.20	0.83	0.29	0.14	0.13	0.06	0.06	0.68	1.51
Sum	1.41	0.40	0.46	2.27	1.10	0.69	0.67	0.06	0.06	2.58	4.85

Table 5.10: Metric M2.1: Relative number of Palladio related problems

	Tool				Methodology								
	Usage	Error	Bug	Sum	Overhead	Distributed system	Projects	Passive Resource	Miscalculation	Probabilities	Solution	Sum	Sum
Media Store													
minor	0.00	0.00	0.11	0.11	0.44	0.00	0.11	0.56	0.11	0.00	0.00	1.22	1.33
intermediate	0.11	0.00	0.00	0.11	0.78	0.44	0.00	0.00	0.11	0.00	0.00	1.33	1.44
major	0.00	0.00	0.00	0.00	0.78	0.33	0.00	0.00	0.00	0.00	0.00	1.11	1.11
Sum	0.11	0.00	0.11	0.22	2.00	0.78	0.11	0.56	0.22	0.00	0.00	3.67	3.89
Web Server													
minor	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.13	0.13	0.00	0.88	0.88
intermediate	0.13	0.00	0.00	0.13	1.88	0.75	0.00	0.00	0.00	0.38	0.13	3.13	3.25
major	0.13	0.00	0.00	0.13	0.13	0.00	0.13	0.00	0.00	0.50	0.00	0.75	0.88
Sum	0.25	0.00	0.00	0.25	2.00	0.75	0.13	0.63	0.13	1.00	0.13	4.75	5.00
Both systems													
minor	0.00	0.00	0.06	0.06	0.22	0.00	0.06	0.59	0.12	0.06	0.00	1.05	1.10
intermediate	0.12	0.00	0.00	0.12	1.33	0.60	0.00	0.00	0.06	0.19	0.06	2.23	2.35
major	0.06	0.00	0.00	0.06	0.45	0.17	0.06	0.00	0.00	0.25	0.00	0.93	0.99
Sum	0.18	0.00	0.06	0.24	2.00	0.76	0.12	0.59	0.17	0.50	0.06	4.21	4.44

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.

		Palladio		SPE	
	Task	Tool	Methodology	Tool	Methodology
Question protocol	0.38	2.13	1.61	0.06	1.29
Acceptance test	0.31	0.06	0.50	0.06	1.08
Error in models	0.75	0.07	0.47	0.11	1.84

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 comprehension, 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.

	SPE-ED	Palladio
Media Store	0.13	0.04
Web Server	0.17	0.10
Average	0.15	0.07

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.

	SPE-ED	Palladio
Media Store	0	0
Web Server	0.06	0.07
Average	0.03	0.05

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

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

Table 5.15: M2.4 Subjective evaluation of the comprehensibility of the Palladio concepts

Concept	Average grade	Standard deviation
Division in scenarios	0.63	1.30
Software model	1.11	0.76
Overhead matrix	0.42	1.26
System model	0.63	1.16
Distributed systems	-0.53	1.26
Different solutions	0.00	1.14
Visualisation of the results	0.16	1.34

Table 5.16: M2.4 Subjective evaluation of the comprehensibility of the SPE concepts

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 participants. 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.

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

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

Advantages	Disadvantages
Easy to use	Inflexible
Fast initial modelling	Less systematic
	Imprecise
	Simulation leads to strange results
	GUI is old-fashioned
	Bugs
	Only for small designs
	Hard to understand modelling of design alternatives

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.

	Whole task		Avg	Initial system		Avg	Avg
	MS	WS		MS	WS		
d_{Pal}	374	285	329.5	203	191	197	263
d_{SPE}	284	243	263.5	99	119	109	186
$\frac{d_{Pal}}{d_{SPE}}$	1.32	1.17	1.25	2.05	1.61	1.81	1.41

Table 5.19: Metric M3.1: Duration of making a prediction

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.

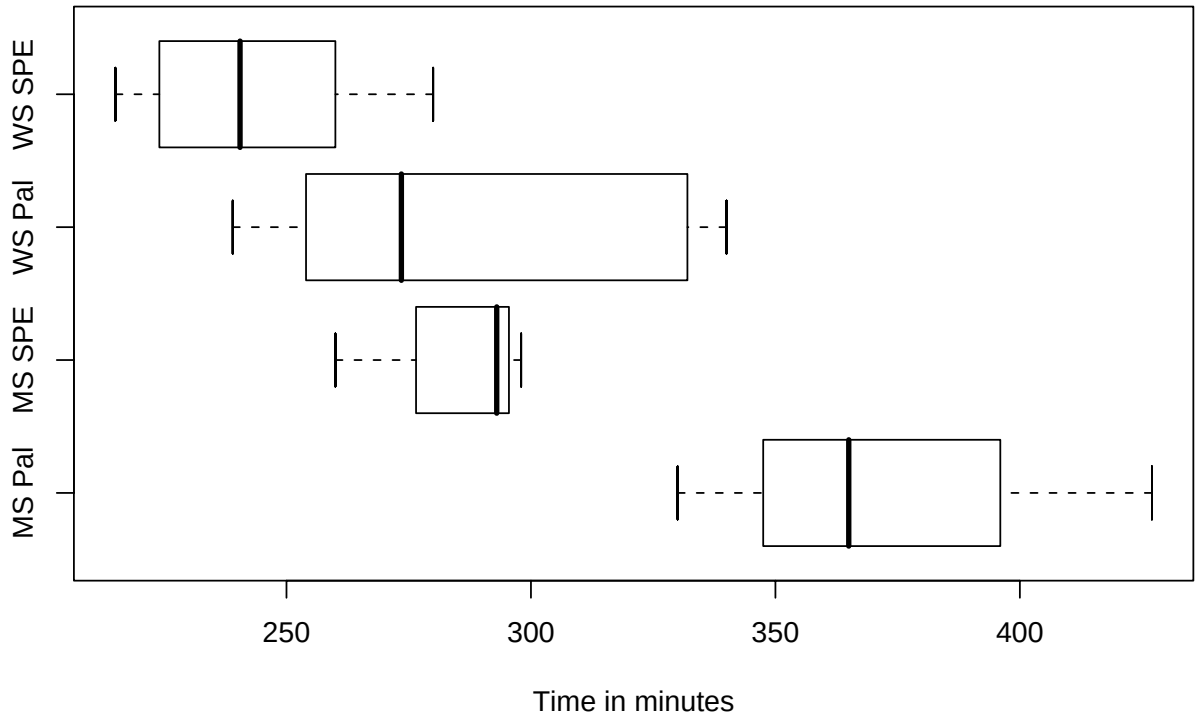


Figure 5.1: Metric M3.1: Duration of whole task for both approaches and both systems

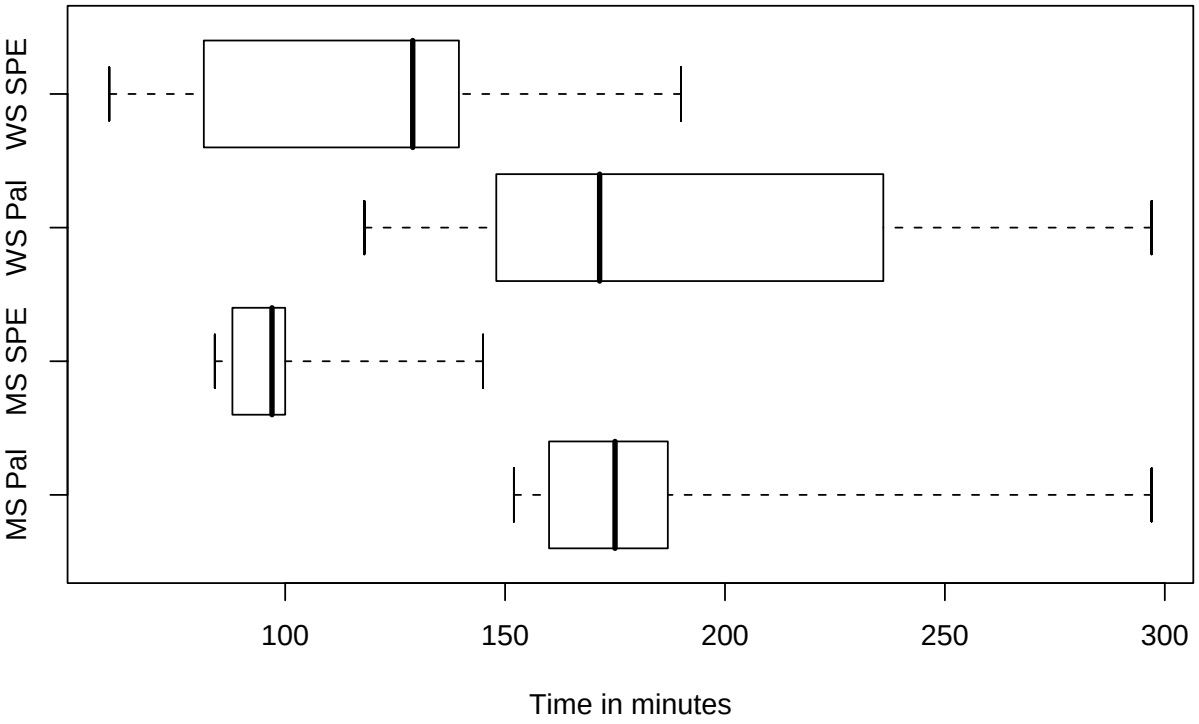


Figure 5.2: Metric M3.1: Duration of the initial system only, for both approaches and both systems

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 μ for the durations differs as follows: $\mu_{Pal} = 1.5\mu_{SPE}$. 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: $\mu_{Pal} > 1.5\mu_{SPE}$. Here, the resulting p-value was 0.923. For the alternative hypothesis that $\mu_{Pal} < 1.5\mu_{SPE}$, 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.

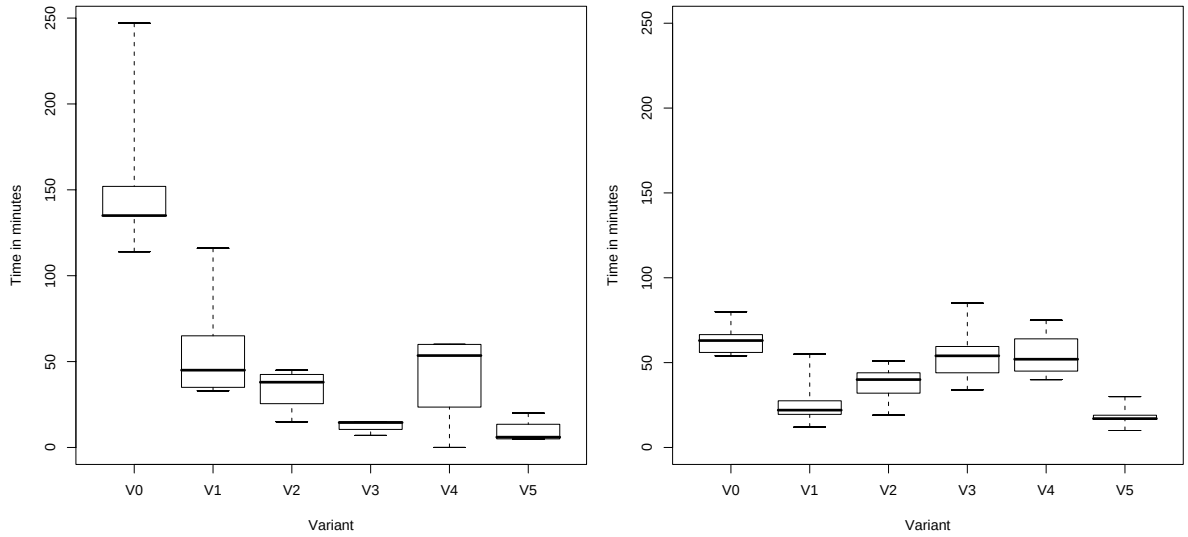
First, I look at the duration of the predictions for the different variants of the system. Generally, I only look at participants who completed the predictions of all variants, because of the aforementioned reasons. I include all participants that provided the duration for the variants, even if they 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).

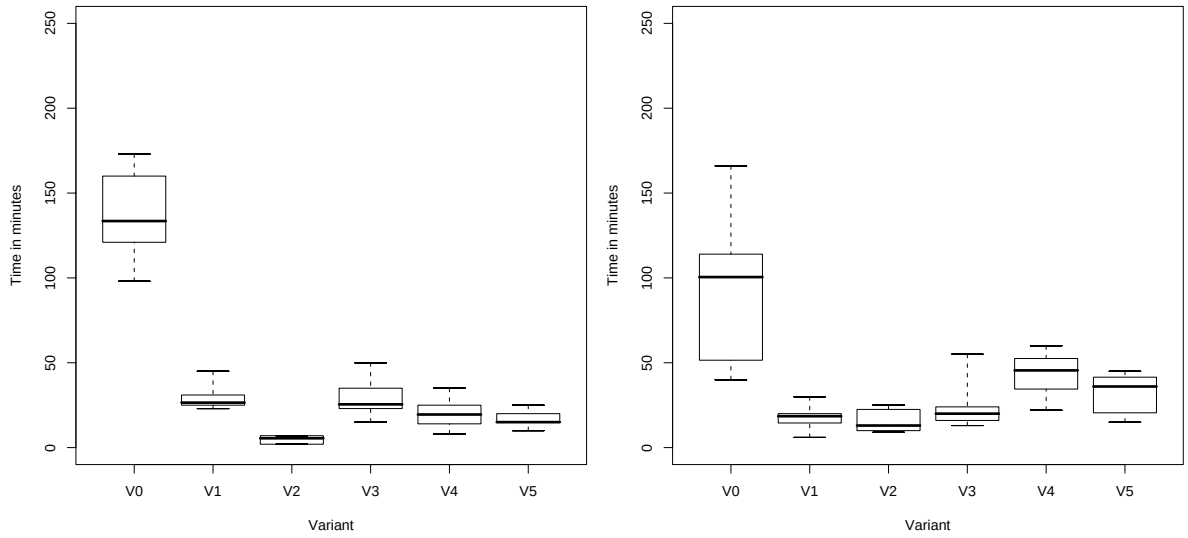
For the Web Server and Palladio, the two participants p_{11} and p_{15} did not finish within the extended time constraints and their results were not included. For the Web Server and SPE, all results were included. The two resulting box plots are shown in figure 5.4, with Palladio on the left-hand side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



(a) Palladio (b) SPE

Figure 5.3: Metric M4.1: Duration of the prediction for the different variants of Media Store



(a) Palladio (b) SPE

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

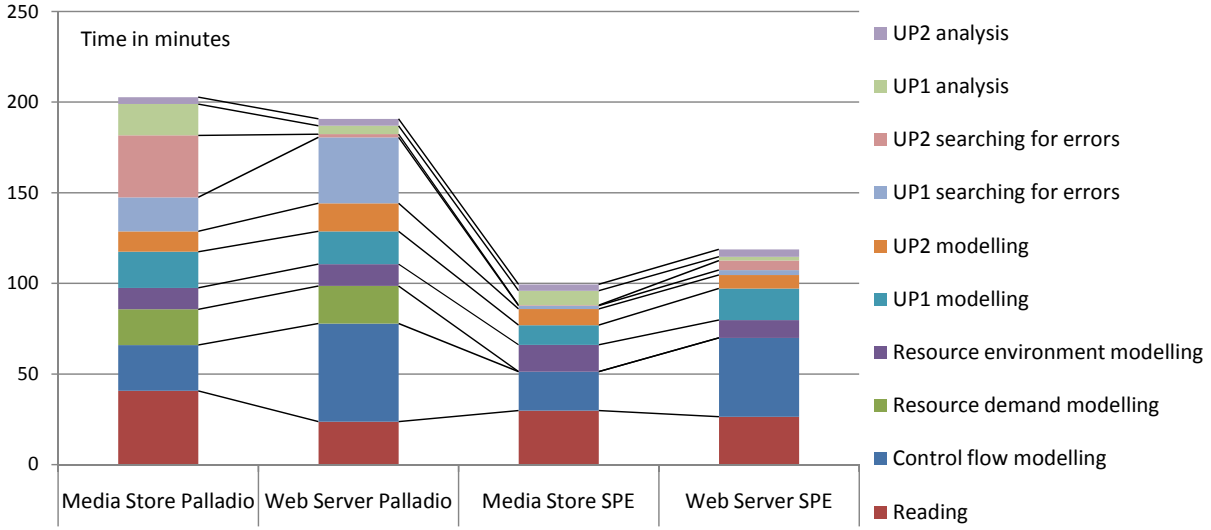


Figure 5.5: Metric M4.2: Break down of the duration for the initial system

	Palladio	SPE
v_0	Modelling, Searching for errors	Modelling
v_1	Modelling, Analysis	Modelling
v_2	Modelling	Modelling
v_3	Modelling, Analysis	Modelling, Searching for errors
v_4	Modelling	Modelling
v_5	Analysis	Modelling

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.

	Palladio	SPE
v_0	Modelling, Searching for errors	Modelling
v_1	Modelling	Modelling
v_2	Modelling	Modelling
v_3	Modelling	Modelling, Searching for errors
v_4	Modelling	Modelling
v_5	Modelling	Modelling UP2

Table 5.21: Metric M4.2: Main time consuming activities for the Web Server

Advantages (No of participants)	**Disadvantages (No of participants)**
Easy to model (5)	Not so precise (1)
Fast (5)	Loss of control (1)
Transparency during modelling (2)	Less flexible (3)
Elegant (1)	Not applicable for complex cases (1)
Not manually (1)	Unclear what happens (1)
	Less practical relevance (1)

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

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.

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_1^{MS}) and the change of the bit rate (v_4^{MS}). Here, v_1^{MS} had a lower average duration but also a higher maximum. For SPE, the introduction of a second server (v_3^{MS}) and the change of the bit rate (v_4^{MS}) were the most time consuming, with v_3^{MS} 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_4^{MS} 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_3^{MS}) 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_1^{MS}) 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_2^{MS}), which was less time-consuming to analyse. Maybe v_1^{MS} 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 M1.2. Then, I consider the results of the metrics for the reasons of the quality specifically targeting Palladio specifics.

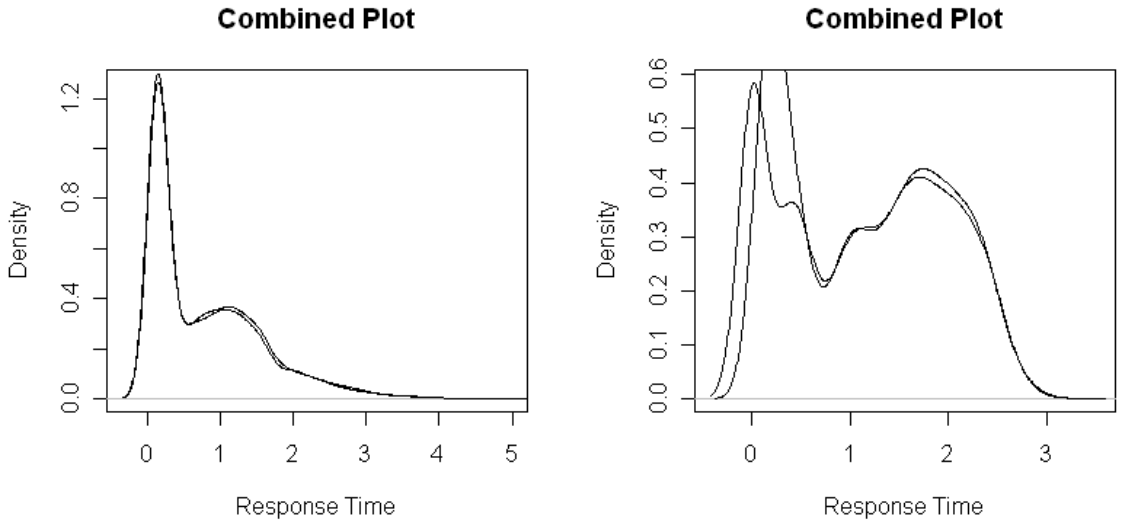
The analysis of the predicted distribution function of Palladio shows that only few distributions passed the K-S test (metric M1.2), and therefore that most distribution probably result from a different distribution function than the one underlying the prototypical distribution. However, the applicability 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 M1.1).

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,



(a) p-value of $4 \cdot 10^{-5}$

(b) p-value of 0

Figure 5.6: Example distributions with K-S test p-value of $4 \cdot 10^{-5}$ and 0

24% pass the test at a 5% significance level. Thus, 24% of the models create were probably identical to the reference model (exept 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 where 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 problem 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

List of Tables

Bibliography

[ABC+06] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 2006.

[BCdK07] Egor Bondarev, Michel R. V. Chaudron, and Erwin A. de Kock. Exploring performance trade-offs of a jpeg decoder using the deepcompass framework. In *WOSP '07: Proceedings of the 6th international workshop on Software and performance*, pages 153–163, New York, NY, USA, 2007. ACM Press.

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The Goal Question Metric Approach. In John J. Marciniak, editor, *Encyclopedia of Software Engineering - 2 Volume Set*, pages 528–532. John Wiley & Sons, 1994.

[BdWC04] Egor Bondarev, Peter H. N. de With, and Michel Chaudron. Predicting Real-Time Properties of Component-Based Applications. In *Proc. of RTCSA*, 2004.

[Bec08] Steffen Becker. *Simultaneous Model Transformations for QoS Enabled Component Based Software Design*. PhD thesis, University of Oldenburg, Germany, 2008. to appear.

[BFG+04] Steffen Becker, Viktoria Firus, Simon Giesecke, Wilhelm Hasselbring, Sven Overhage, and Ralf Reussner. Towards a Generic Framework for Evaluating Component-Based Software Architectures. In Klaus Turowski, editor, *Architekturen, Komponenten, Anwendungen - Proceedings zur 1. Verbundtagung Architekturen, Komponenten, Anwendungen (AKA 2004), Universität Augsburg*, volume 57 of *GI-Edition of Lecture Notes in Informatics*, pages 163–180. Bonner Köllen Verlag, December 2004.

[BGdMT98] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons Inc., 1998.

[BGMO06] Steffen Becker, Lars Grunske, Raffaela Mirandola, and Sven Overhage. Performance Prediction of Component-Based Systems: A Survey from an Engineering Perspective. In Ralf Reussner, Judith Stafford, and Clemens Szyperski, editors, *Architecting Systems with Trustworthy Components*, volume 3938 of *LNCS*, pages 169–192. Springer, 2006.

[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-

ferent performance prediction methods to distributed enterprise applications. *J. Supercomput.*, 34(2):93–111, 2005.

[BJHN04] D. A. Bacigalupo, S. A. Jarvis, L. He, and G. R. Nudd. An investigation into the application of different performance techniques to e-commerce applications. In *18th IEEE International Parallel and Distributed Processing Symposium 2004 (IPDPS'04), Santa Fe, New Mexico*. IEEE Computer Society Press, April 2004.

[BK96] Falko Bause and Pieter S. Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory*. Vieweg-Verlag, 1996.

[BKR07a] Steffen Becker, Heiko Koziolek, and Ralf Reussner. Model-based Performance Prediction with the Palladio Component Model. In *Proceedings of the 6th International Workshop on Software and Performance (WOSP2007)*. ACM Sigsoft, February5–8 2007.

[BKR07b] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio Component Model for Model-Driven Performance Prediction. *Journal of Systems and Software*, 2007. to appear.

[BM03a] Simonetta Balsamo and Moreno Marzolla. A Simulation-Based Approach to Software Performance Modeling. In *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 363–366. ACM Press, 2003.

[BM03b] A. Bertolino and R. Mirandola. Towards component based software performance engineering. In Ivica Crnkovic, Heinz Schmidt, Judith Stafford, and Kurt Wallnau, editors, *Proc. 6th Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction, ACM/IEEE 25th International Conference on Software Engineering ICSE 2003*, pages 1–6, 2003.

[BM04] Antonia Bertolino and Raffaela Mirandola. CB-SPE Tool: Putting Component-Based Performance Engineering into Practice. In Ivica Crnkovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt C. Wallnau, editors, *Proc. 7th International Symposium on Component-Based Software Engineering (CBSE 2004), Edinburgh, UK*, volume 3054 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2004.

[BMDI04] Simonetta Balsamo, Moreno Marzolla, Antinisca Di Marco, and Paola Inverardi. Experimenting different software architectures performance techniques: a case study. In *Proceedings of the fourth international workshop on Software and performance*, pages 115–119. ACM Press, 2004.

[BMIS04] Simonetta Balsamo, Antinisca Di Marco, Paola Inverardi, and Marta Simeoni. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, May 2004.

[BR06] Steffen Becker and Ralf Reussner. The Impact of Software Component Adaptation on Quality of Service Properties. *Lóbjet*, 12(1):105–125, 2006.

[BTJN03] D. A. Bacigalupo, J. D. Turner, S. A. Jarvis, and G. R. Nudd. A dynamic predictive framework for e-business workload management. In *7th World Multiconference on Systemics,Cybernetics and Informatics (SCI2003) Performance of Web Services Invited Session, Orlando, Florida, USA*, July 2003.

[CC79] Thomas D. Cook and Donald Thomas Campbell. *Quasi-experimentation design & analysis issues for field settings*. Houghton Mifflin, Boston, 1979.

[CGLL02] Shiping Chen, Ian Gorton, Anna Liu, and Yan Liu. Performance Prediction of COTS Component-based Enterprise Applications. In Ivica Crnkovic, Heinz Schmidt, Judith Stafford, and Kurt Wallnau, editors, *Proceeding of the 5th ICSE Workshop on Component-Based Software Engineering: Benchmarks for Predictable Assembly*, Orlando, Florida, May 2002.

[CM02] Vittorio Cortellessa and Raffaela Mirandola. PRIMA-UML: a performance validation incremental methodology on early UML diagrams. *Sci. Comput. Program*, 44(1):101–129, 2002.

[Dal03] Peter Dalgaard. *Introductory statistics with R*. Springer-Verlag, New York, USA, corr. print. edition, 2003.

[DAL04] Jozo Dujmović, Virgilio Almeida, and Doug Lea, editors. *WOSP '04: Proceedings of the 4th international workshop on Software and performance*, New York, NY, USA, 2004. ACM Press. General Chair Jozo Dujmović and Program Chair Virgilio Almeida and Program Chair-Doug Lea.

[Dal07] Peter Dalgaard. Power calculations for one and two sample t tests. `http://stat.ethz.ch/R-manual/R-patched/library/stats/html/power.t.test.html`, 2007. last retrieved 2007-11-12.

[DES07] The DESMO-J Homepage, 2007.

[DHL96] Christiane Differding, Barbara Hoisl, and Christopher M. Lott. Technoloy package for the Goal Question Metric Paradigm. Technical report, University of Kaiserslautern, AG Software Engineering, 1996.

[DRSS01] Reiner R. Dumke, Claus Rautenstrauch, Andreas Schmietendorf, and André Scholz, editors. *Performance Engineering, State of the Art and Current Trends*, volume 2047 of *Lecture Notes in Computer Science*. Springer, 2001.

[Ecla] Eclipse Foundation. Eclipse modeling framework homepage. `http://www.eclipse.org/modeling/emf/`. last retrieved 2007-10-24.

[Eclb] Eclipse Foundation. Graphical modeling framework homepage. `http://www.eclipse.org/gmf/`. last retrieved 2007-11-11.

[ER03] Albert Endres and Dieter Rombach. *A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories*. Addison-Wesley, Reading, MA, USA, 2003.

[FB04] Viktoria Firus and Steffen Becker. Towards performance evaluation of component-based software architectures. In *Formal Foundations of Embedded*

Software and Component-based Software Architectures (FESCA), pages 118–121. ETAPS 2004, 2004.

[FBH05] Viktoria Firus, Steffen Becker, and Jens Happe. Parametric Performance Contracts for QML-specified Software Components. In *Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA)*, volume 141 of *Electronic Notes in Theoretical Computer Science*, pages 73–90. ETAPS 2005, 2005.

[FER08] Felix Freiling, Irene Eusgeld, and Ralf Reussner, editors. *Dependability Metrics*. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2008. to appear.

[FKB+05] Viktoria Firus, Heiko Koziolek, Steffen Becker, Ralf Reussner, and Wilhelm Hasselbring. Empirische Bewertung von Performanz-Vorhersageverfahren fr Software-Architekturen. In Peter Liggesmeyer, Klaus Pohl, and Michael Goedicke, editors, *Software Engineering 2005 Proceedings - Fachtagung des GI-Fachbereichs Softwaretechnik*, volume 64 of *GI-Edition of Lecture Notes in Informatics*, pages 55–66. Bonner Köllen Verlag, March 2005.

[Gil07] D Gilbert. Jfreechart, 2007.

[Gla98] Robert L. Glass. *Software Runaways: Monumental Software Disasters*. Prentice Hall, Englewood Cliffs, NJ, USA, 1998.

[GMS05] Vincenzo Grassi, Raffaela Mirandola, and Antonino Sabetta. From Design to Analysis Models: a Kernel Language for Performance and Reliability Analysis of Component-based Systems. In *WOSP '05: Proceedings of the 5th international workshop on Software and performance*, pages 25–36, New York, NY, USA, 2005. ACM Press.

[GR98] Rob Guth and Lynda Radosevich. IBM crosses the Olympic finish line. http://www.infoworld.com/cgi-bin/displayStory.pl?/features/980209olympics.htm, February 1998. accessed August 1st, 2007.

[GVR02] Robert L. Glass, Iris Vessey, and Venkataraman Ramesh. Research in software engineering: an analysis of the literature. *Information & Software Technology*, 44(8):491–506, 2002.

[HHK02] Holger Hermanns, Ulrich Herzog, and Joost-Pieter Katoen. Process Algebra for Performance Evaluation. *Theoretical Computer Science*, 274(1–2):43–87, 2002.

[HM07] Frank Heitmann and Daniel Moldt. Petri nets tool database, 2007.

[HMSW02] Scott A. Hissam, Gabriel A. Moreno, Judith A. Stafford, and Kurt C. Wallnau. Packaging Predictable Assembly. In Judy M. Bishop, editor, *Component Deployment, IFIP/ACM Working Conference, CD 2002, Berlin, Germany, June 20-21, 2002, Proceedings*, volume 2370 of *Lecture Notes in Computer Science*, pages 108–124. Springer, 2002.

[IEE90] IEEE Standards Board. IEEE standard glossary of software engineering terminology—IEEE std 610.12-1990, 1990.

[IS04] James Ivers and Natasha Sharygina. Overview of comfort: A model checking reasoning framework. Technical report, Software Engineering Institute, Carnegie Mellon University, April 2004.

[KBH07] Heiko Koziolek, Steffen Becker, and Jens Happe. Predicting the performance of component-based software architectures with different usage profiles. In *Proceedings of the 3rd International Conference on the Quality of Software Architectures (QoSA)*, volume To appear of *LNCS*, Boston, USA, Juli 2007.

[KHB06] Heiko Koziolek, Jens Happe, and Steffen Becker. Parameter dependent performance specification of software components. In *Proceedings of the Second International Conference on Quality of Software Architectures (QoSA2006)*, volume 4214 of *Lecture Notes in Computer Science*, pages 163–179. Springer-Verlag, Berlin, Germany, July 2006.

[Koz04] Heiko Koziolek. Empirische Bewertung von Performance-Analyseverfahren für Software-Architekturen. Master's thesis, Universität Oldenburg, 2004.

[MAD94] Daniel Menasce, Virgilio Almeida, and Larry Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*. Prentice-Hall, New Jersey, March 1994. $44.

[Mar04] Moreno Marzolla. *Simulation-Based Performance Modeling of UML Software Architectures*. PhD Thesis TD-2004-1, Dipartimento di Informatica, Università Ca' Foscari di Venezia, Mestre, Italy, February 2004.

[Mar05] Anne Martens. Empirical Validation and Comparison of the Model-Driven Performance Prediction Techniques of CB-SPE and Palladio. study thesis, University of Oldenburg, 2005.

[MK00] Ramiro Montealegre and Mark Keil. De-escalating Information Technology Projects: Lessons from the Denver International Airport. *MIS Quarterly*, 3:417–447, 2000.

[MTW03] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, 8/18, 2003.

[Obj05] Object Management Group (OMG). UML Profile for Schedulability, Performance and Time. http://www.omg.org/cgi-bin/doc?formal/2005-01-02, January 2005.

[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.

[Per03] Performance Engineering Services, Austin, TX. *SPE-ED User Guide*, 2003. http://www.perfeng.com.

[Pou96] Jeffrey S. Poulin. *Measuring software reuse: principles, practices, and economic models*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[Pre01] Lutz Prechelt. *Kontrollierte Experimente in der Softwaretechnik*. Springer-Verlag, Berlin, Germany, 2001.

[RBK+07] Ralf H. Reussner, Steffen Becker, Heiko Koziolek, Jens Happe, Michael Kuperberg, and Klaus Krogmann. The Palladio Component Model. Interner Bericht 2007-21, Universität Karlsruhe (TH), Faculty for Informatics, Karlsruhe, Germany, October 2007.

[RH06] Ralf H. Reussner and Wilhelm Hasselbring. *Handbuch der Software-Architektur*. dPunkt.verlag, Heidelberg, 2006.

[Sac97] Lothar Sachs. *Angewandte Statistik: Anwendung statistischer Methoden*. Springer-Verlag, Berlin, Germany, 8., völlig neu bearb. und erw edition, 1997.

[SDJ07] Dag I. K. Sjøberg, Tore Dybå, and Magne Jørgensen. The future of empirical methods in software engineering research. *fose*, pages 358–378, 2007.

[SKK+01] Murali Sitaraman, Greg Kuczycki, Joan Krone, William F. Ogden, and A.L.N. Reddy. Performance Specification of Software Components. In *Proceedings of the 2001 symposium on Software reusability: putting software reuse in context*, pages 3–10. ACM Press, 2001.

[SLC+05] Connie U. Smith, Catalina M. Llado, Vittorio Cortellessa, Antinisca Di Marco, and Lloyd G. Williams. From UML Models to Software Performance Results: an SPE Process based on XML Interchange Formats. In *WOSP '05: Proceedings of the 5th international workshop on Software and performance*, pages 87–98, New York, NY, USA, 2005. ACM Press.

[Smi90] Connie U. Smith. *Performance Engineering of Software Systems*. Addison-Wesley, Reading, MA, USA, 1990.

[SOB01] Dennis Smith, Liam O'Brien, and John Bergey. Mining components for a software architecture and a product line: the options analysis for reengineering (OAR) method. In *Proceedings of the 23rd international conference on Software engineering*, page 728. IEEE Computer Society, 2001.

[SS01] A. Schmietendorf and A. Scholz. Aspects of Performance Engineering - an Overview. In R. et. al. Dumke, editor, *Performance Engineering: State of the art and current trends*, volume 2047 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 2001.

[SW93] C. U. Smith and L. G. Williams. Software performance engineering: A case study including performance comparison with design alternatives. *IEEE Trans. Softw. Eng.*, 19(7):720–741, 1993.

[SW97] Connie Smith and Lloyd Williams. Performance engineering evaluation of object-oriented systems with SPEED. 1245, 1997.

[SW98] Connie U. Smith and Lloyd G. Williams. Performance engineering evaluation of CORBA-based distributed systems with SPE•ED. *Lecture Notes in Computer Science*, 1469:321–??, 1998.

[SW02] C. U. Smith and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley, 2002.

[Szy98] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley, Reading, MA, USA, 1998.

[TLPH95] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28(1):9–18, January 1995.

[Wel47] B. L. Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 34:28–35, 1947.

[WMW03] Xiuping Wu, David McMullan, and Murray Woodside. Component Based Performance Prediction. In *6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*, pages 13–18, Portland, Oregon, May 2003.

[WPS+05] Murray Woodside, Dorina C. Petriu, Hui Shen, Toqeer Israr, and Jose Merseguer. Performance by unified model analysis (PUMA). In *WOSP '05: Proceedings of the 5th International Workshop on Software and Performance*, pages 1–12, New York, NY, USA, 2005. ACM Press.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: an Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[WW04] Xiuping Wu and Murray Woodside. Performance Modeling from Software Components. *SIGSOFT Softw. Eng. Notes*, 29(1):290–301, 2004.

A Tutorial Slides and Preparatory Exercises

A.1 Introductory Tutorial Slides

The organisational and basics slides were created by Klaus Krogmann, the tool session slides by the author.

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßiger Software-Entwurf

Organisatorisches & Einführung

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Überblick

- Motivation
- Themen
- Organisation

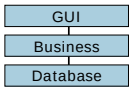
Motivation

Goals

- Why do I sit here?
- Learning to critically evaluate:
  - middleware platforms
  - development tools
  - component-based designs
  - Therefore: One need to know the principles and what is possible, not just what currently exists.
- Scientific Background of component-based software engineering
- Evaluation of design decisions at an architectural level

Design Alternatives (1)

- Given
  - A three-tier architecture
  - Each layers runs on an own machine with fixed hardware
  - Fixed, but limited network bandwidth



- Question: Is it worth compressing data between
  - GUI and Business
  - Business and Database?
- Pro: Less data to transfer
- Contra: Increased CPU load

Design Alternatives (2)

- Further information is needed to decide
  - Load of each layer by simultaneous user access
  - Design and timing behavior of each layer
  - Kind and amount of data to transfer
  - …
- Still, the question remains: Is it worth compressing data?
- No intuitive answer available
- **Solution**: Systematic approach to support design decisions

Themenüberblick

Themen des Praktikums

Themen des Praktikums

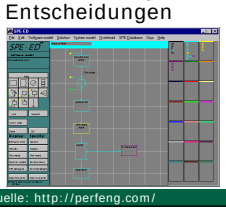
- Systematischer Entwurf von Software-Systemen
- Einsatz komponentenbasierter Technologien
- Blick auf nicht-funktionale Eigenschaften
  - Im Besonderen: Performanz
  - Verwendung modellgetriebener Vorhersageverfahren
- Verfahren
  - Software Performance Engineering (SPE)
  - Palladio

Werkzeuge

- Vorhersage- und Bewertungsverfahren werden durch Werkzeuge unterstützt

Werkzeug: SPE·ED

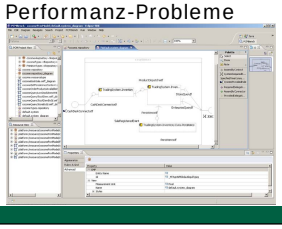
- Autor: Connie U. Smith
- Erstellen und lösen von visuell repräsentierten Performanz-Modellen
- Unterstützung von u. a. Entwurfs-Entscheidungen



Quelle: http://perfeng.com/

Werkzeug: Palladio

- Unterstützung eines gesamten komponentenbasierten Entwurfsprozesses
- Analyseverfahren liefern Hinweise auf Performanz-Probleme



Organisation

Aktuelle Informationen

- Aktuelle Informationen und Ankündigungen in unserem Wiki:

  http://sdqweb.ipd.uka.de/wiki/

  Praktikum Ingenieurmäßiger Software-Entwurf SS07

- Zugang zu Materialien
  - Benutzer: stud
  - Passwort: ise07

Die Veranstaltung

- Mix aus
  - Theoretischer Vorbereitung in Vorlesungen
  - Bewerteten Übungsblättern als Hausübung
  - Kleineren Tests zur Kontrolle des Lernfortschritts
  - Praktische Übungen im Poolraum
    - Selbstständige Übungen
    - Betreute Übungen
  - Praktischer Einsatz moderner Entwurfswerkzeuge
  - Große Experimentsitzung am Ende des Semesters

Rahmen

- 13 Wochen
- 2 SWS Theorie
- 2 SWS
  - Übungsblatt-Besprechung
  - Praxis-Stunden im Poolraum
    - Lösung von Übungsblättern
    - Tutoren stehen als Ansprechpartner zur Verfügung
- Nicht in jeder Woche finden alle Veranstaltungsformen statt (→ Terminübersicht)

Regelmäßige Termine

- Mittwochs: Theorie (SR 301)
  - 17:30 bis 19:00 Uhr
- Freitags: Praxis (Raum 356)
  - 9:45 bis 11:30 Uhr
  - Weitere Poolraum-Nutzung („exklusiv")
    - Mittwochs von 14:00 bis 17:30 Uhr
    - Freitags von 9:45 - 11:30 Uhr
    - Zugang → später

Experiment Termine

- Das große Experiment findet an zwei Samstagen statt:
  - **30. Juni 2007**
  - **07. Juli 2007**
  - Ganztägig
- Größere Beispiel-Systeme werden mit zwei Entwurfs-Verfahren behandelt
  → Praktische Anwendung des Gelernten

\+ Freies Essen & Getränke 

Termine

1. Informationsveranstaltung, Orga, Einführung
2. KBSE: Grundlagen: Was sind Komponenten, Überblick PCM-Rollen
3. Einführung in die Werkzeuge (Eclipse, SVN, …)
4. SPE
5. Schätzen nicht-funktionaler Eigenschaften
6. Komponentenentwickler-Session / SEFF-1-Session
7. SEFF-2-Session
8. System-Architekt-Session
9. Deployer-Session / QoS-Analyst
10. Domain-Expert / Usage Model
11. 1. Experimentsitzung
12. 2. Experimentsitzung
13. Wrap-Up

Übungen

- Übungsblätter und Tutorien werden zu wechselnden 2er-Gruppen durchgeführt
- Einteilung erfolgt durch uns
- Übungsblätter werden in Tutorien besprochen

Bewertung

- Für die Praktikumsleistung wird eine Note vergeben
- Grundlage
  - Bearbeitung von Übungsblättern
  - Individual-Prüfungen (für alle Teilnehmer) in Form von schriftlichen Kurztests zu Beginn von Praktikumssitzungen: Abfrage des Lernfortschritts
  - Regelmäßige Teilnahme
  - Güte der erstellten Vorhersagemodelle

Regeln

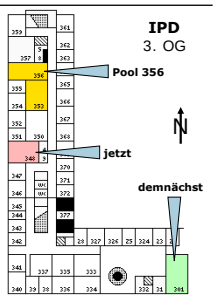
- Um erfolgreich das Praktikum bestehen zu können, dürfen Sie maximal an zwei Terminen (Vorlesungen, Übungen, Kurztests) fehlen. Verpasste Kurztests werden nachgeholt.
- Für die beiden Experimentsitzungen besteht Anwesenheitspflicht (Ausnahme: rechtzeitig eingereichtes ärztliches Attest)
- Es darf maximal ein Übungsblatt als „nicht bearbeitet" gewertet worden sein. → Details auf dem ersten Übungsblatt

Aufnahmetest

- Am Freitag, 27. April findet ein Aufnahmetest statt
- Themen:
  - Prüfung der Kenntnisse aus der Softwaretechnik-Vorlesung
    - UML
    - Modellierung im Allgemeinen
  - Grundlagen der komponentenbasierten Software-Entwicklung
  - Komponentenbasierter Entwicklungsprozess
- Bestehen für weitere Teilnahme am Praktikum notwendig

Poolraum

- **SWT-Poolraum 356**, Geb. 50.34, Informatik-Hauptgebäude, 3. OG
- Eigene Notebooks dürfen genutzt würden
- Rechner in Raum 353 als Ausweichmöglichkeit
- Zugangsdaten werden ausgegeben



Empfehlung

- Vorlesung "**Komponentenbasierte Software-Entwicklung**" eignet sich hervorragend als Ergänzung zum Praktikum.
- **Montags** von **14:00 - 15:30 Uhr** Raum: HS -101 (50.34 UG)

Vorlesung KBSE

- Theoretische Inhalte werden vertieft
- Blick auf
  - Prozesse
  - (Komponenten-) Architektur-Muster
  - Komponenten-Entwurf
  - Komponententechnologien u. a. EJB

  uvm.

Universität Karlsruhe (TH)
Research University ▪ founded 1825

Laboratory: Software Design as an Engineering Discipline

Basics:
Components & Roles in
Component Based Development Process

Ralf Reussner (reussner@ipd.uka.de)

Chair Software Design and Quality
Institute for Program Structures and Data Organization (IPD)
Faculty of Informatics, Universität Karlsruhe (TH)

Overview

- Component definitions
- Engineering approach
- Component basics

Introduction into Component Based Software Development

Compare for Lecture:
"Komponentenbasierte Software-Entwicklung"

Components (1)

- Not a new idea! (McIlroy 1968, NATO-Conference Garmisch-Patenkirchen)
- Anything but reusable [HC91]
- undefinable (natural concept) [CE00]
  - opposed to artificial concept, like objects
- Building blocks for software
- Software: Code or all artefacts?

Components (2)

- Properties of components [Szyperski, 1997, p. 30]:
  - unit of independent deployment
  - unit of third party composition
  - has no persistent state
- Consequences:
  - software components (for C. Szyperski): executable code
  - independent deployment: well-documented requires-interfaces
  - no persistent state: component cannot be distinguished from a copy.

Components (3)

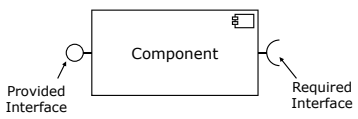
- A components is an artefact of the software development process with a description of its application. [Goos 00]
  - all artefacts
  - deployment may require additional effort (generation, compilation, binding, ...)

This is a component...

- A component is a contractually specified building block for software which can be readily composed by third parties without understanding its internal structure. (Reussner)
  - Not necessarily black-box: information on component's internals may be provided for tools
  - Readily composed: effort for deployment, assembly, composition or adaptation should be as transparent as possible.
- "Components are for composition, much beyond is unclear..." (Clemens Szyperski)

UML2: Components and Interfaces

- A short introduction into the UML2 notation of components



Roots of Component Systems

- Distributed Object Systems (CORBA, etc.)
- Embedded Document Standards (OLE)
- Graphical User Interface Libraries (JavaBeans)
- Programming-in-the-Large [DeRemer & Kron76]

Components in Context

- Objects / Classes
- ADTs
- Modules
  - Module as a manager
  - Module as a type
- **Components just a new buzzword?**

Objects

- Three defining properties of an Object [Booch 1994]
  - Identity
  - State
  - Behaviour (encapsulated with state)

Components and Classes / Objects

- Life-cycle of object is controlled by method invocations at run-time which are programmed at design-time.
- Component deployment context changes after compilation.
- One component may consist out of several classes

Differences between Component and Classes

Component	**Class**
Executable run-time entity	Design-time entity
May contain several classes	Code often required
No code available	Use by inheritance and delegation
Description by Interfaces	Often designed for one system
Only usable by delegation	Deployment context does not change after compilation
Developed separately	
Deployment context changes after compilation	

Components and Abstract Data Types

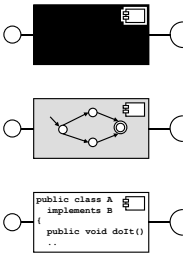
- Abstract data type (ADT) [LZ74]
  - set of types
  - set of signatures (i.e., functions with their return- and parameter type)
  - set of axioms (constraints describing the composition of functions)
- Basically an abstract class plus some semantics by the axioms

Modules / Component

- Similar:
  - Elements for hierarchical system decomposition
  - Used through interface
  - Often only one instance available
- Different:
  - Modules are not contractually specified entities themselves (as objects / classes are not, too)
  - Import-clause of modules is insufficient for a requires interface

Black Box /Grey Box / White Box for Components

- Black Box
  - No internal information
- Grey Box
  - An abstract view of internals
  - E. g. specification (not realization) of internal behaviour
- White Box
  - All information about a component is given
  - E. g. source code

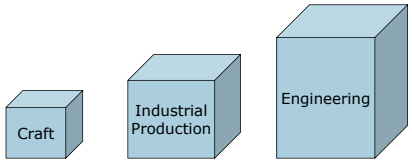


Component-Specific Benefits (1)

- Ease of (run-time) reconfiguration:
  - software families (evolution)
  - software product lines (variation)
- Efficiency:
  - saving memory by hot-swapping required components into memory while removing unused ones. (embedded consumer electronics, etc.)
  - (note lower speed by more explicit interfaces, not removed by compilers)

Component-Specific Benefits (2)

- Engineering with Components
- What is an engineering discipline?



Craft and Industrialised Production

- Craft
  - customer and developer often same person
  - talent and experience instead of understanding
- Industrial Production
  - specialised personal
  - communication, documentation

Engineering (1)

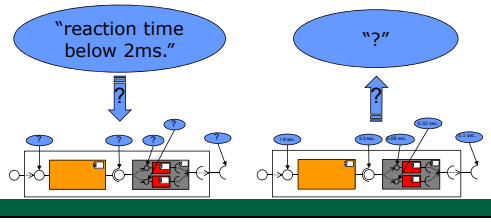
- Consideration of efficiency, costs and time
- Scientific understanding and standards of
  - products
  - processes
- Selection of appropriate tools and processes
- Systematic development of product

Engineering (2)

- Systematic Treatment of Quality Attributes

Decomposition of global System-Requirements

Prediction of global System-Properties



Engineering (3)

- Systematic construction
  - selecting appropriate components according to requirements
  - reasoning on system properties in terms of component properties and designs / architectures
- "correctness-by-construction"

Engineering for Software (1)

- Danger of Analogies:
- Software-IC (McIlroy, Cox)
  - different use of shared resources
  - using less functionality often no problem for hardware
- Software and Cars
  - Software production is a non-issue, while software development is
    (Car development is to be compared with software development)

Engineering for Software (2)

- Probably best analogy:
  Software and Architecture / Building
- each project is an individual development project using existing components.

State of Software Engineering

- "No progress": same problems as for decades
  - Requirements
  - Correctness
  - Performance (quality in general)
  - "Software crisis": "x% of all larger software projects run over time, budget or fail".

No Progress Made?

- The same problems as stated 1968 (first Software Engineering Conference)
- "the problem of achieving sufficient reliability in the data systems..."
- "the difficulties of meeting schedules and specifications on large software projects"
- "the highly controversial question of whether software should be priced separately from hardware"

The Title says it all...

- U.S. Government Accounting Office: "*Contracting for Computer Software Development – Serious Problems Require Management Attention to Avoid Wasting of **Additional** Millions.*"
  Technical Report FFGMSD-80-4, Nov. 1979
- **Failed** projects under investigation!

State of Software Engineering

- Progress: same problems as for decades but for substantially more complex and larger systems.
- „Planning crisis" instead of „Software crisis" [Glass 00]:
  - Project budget and timing is done by customers or managers
  - Only two of the following three should be made by customers / managers [XP]:
    - Functionality, time, budget

Bottom-up / Top down

- **Original view in CBSE:**
  bottom-up development:
  1. Components are developed independently of each other and independently of an application
  2. Applications are created by assembling components
  - **Never became reality!**
- **~2000:** top-down development:
  1. frameworks and software applications with explicit plug-in extension mechanisms are developed
  2. Plug-ins are developed to fill these applications
     Example: web browser, Photoshop, Eclipse, etc.
- **Since ~2004:**
  - New applications are developed and explicitly are designed to make use of existing plug-ins of *other* applications (e.g., Photoshop plug-ins form a de-facto standard for image processing tools)

Lessons Learned

- Understand
  - Why components are used
  - Distinguish the terms components, modules, classes, objects, abstract data type
  - Black-box, grey-box, white-box
  - Benefits of
    - Component use
    - Engineering

Component

Basics

Overview

- UML2 – Short introduction
- Components – Basics
- CBSE process and roles

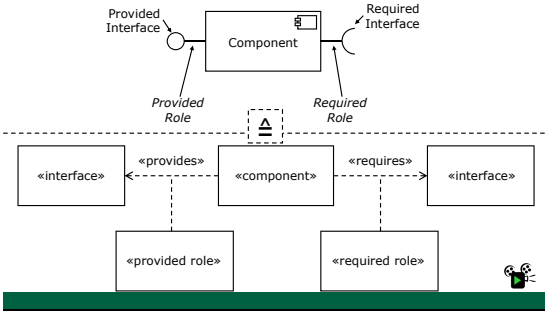
Goals

- Learn the basics of UML2: views supporting component development
- Basics of components
  - Overview on component types
  - Component instance levels
  - Differences between component types and instances
  - Interfaces
  - Behavioral descriptions
- Understand the way the CBSE development process is designed and which tasks have to be done

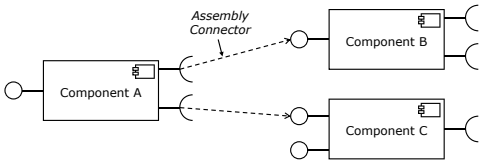
UML2

Short introduction

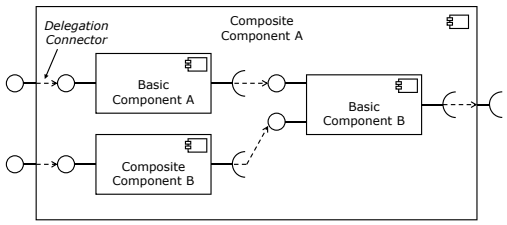
UML2: Abbreviated Notations



UML2: Assembly Connectors

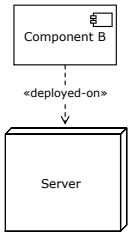


UML2: Composite Components and Delegation Connectors



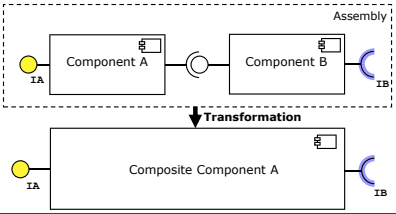
UML2: Deployment

- Creating a deployment instance == putting a implementation instance into a context

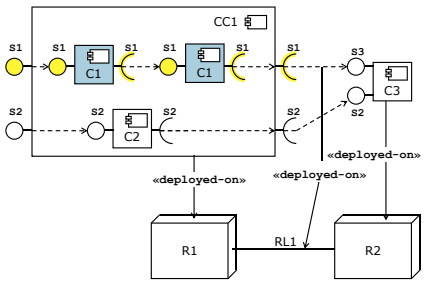


Assembly / Composition

- Bundling of components (with the inner component structure of the bundle invisible/visible)



Notation for static component architecture



Components

Basics (continued)

Components in Detail: Overview

- Components: Types and Instances
- Interfaces
- Static Structures
- Behavioral Description

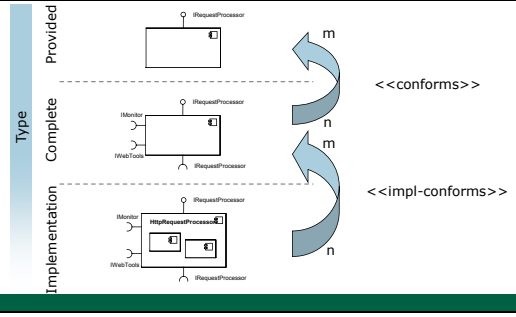
Component

- Characterised by
  - Its roles (provided and required)
  - The component type (next slides)
- Important Difference:
  - Component Type
  - Component Instances

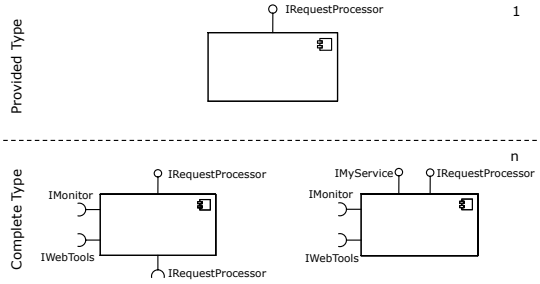
Component Type vs. Component Implementation

- Multiple layers
  - Types occur in multiple forms
    - Provided Component Type
    - Complete Component Type
    - Implementation Component Type
    - → Hierarchy: Next slide
  - Component Instances appear at different levels
    - Instances of implementation types

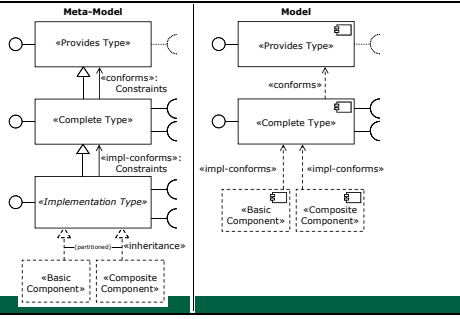
Component Type Hierarchy



Component Types



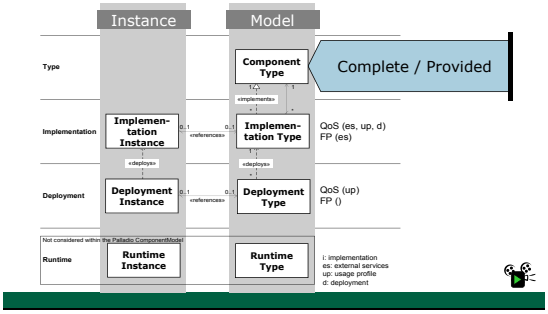
Component Types



Implementation Component Types

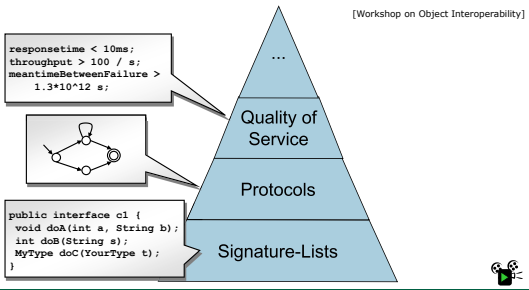
- Do not mix up Implementation Component Types with the implementation itself
- Implementation Component Type is an *abstraction* of a *number* of possible implementations
- In the Palladio world two kinds exist
  - Basic Component: "atoms"
  - Composite Component: "molecules"
  - → More Details follow in later lectures

Term: Component



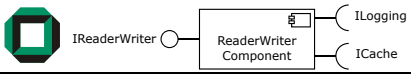
Hierarchy of Interface Models

[Workshop on Object Interoperability]



Signature-List-Based Interfaces

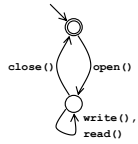
- Unordered list of signatures
- Importance of naming conventions / standards
  - no behavioural information given by signature
- Realised in: Java interfaces, EJB, CORBA-IDL, .NET



Protocols

IReaderWriter

```
Handle open()
   throws SecurityException;
void write(string s, Handle h);
string read(Handle h);
void close(Handle h);
```



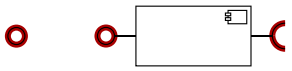
ILogging

```
void writeLog(string log)
   throws WriteException;
```

ICache

```
string readFromCache(Handle h)
   throws CacheMissException;
```

Interface



- Define Services (by signature lists):
  `public void doIt()`
- Can exist by their own ("first class entity")

Signatures

- Signature:

1. access modifier
2. return-type
3. name
4. ordered list of parameter-types and names
5. unordered list of exceptions, possibly thrown

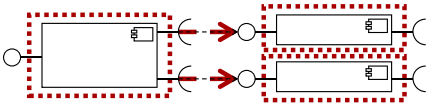
Example:

1. `public`
2. `Customer[]`
3. `getCustomers`
4. `Customer query`
5. `InvalidDB`

Example Interface...

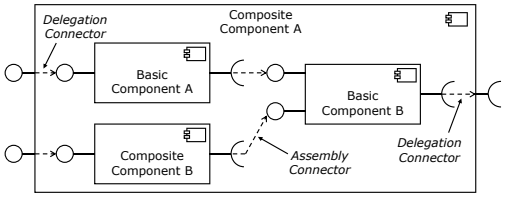
```
providesInterface MemoryMgr {
  int init (int);
  void release();
  int read(int);
  void write(int);
  void layout1();
  void layout2();
  void free();
}
```


Static Structure



- Wiring of Components
- What in fact in connected / wired
- What contexts are

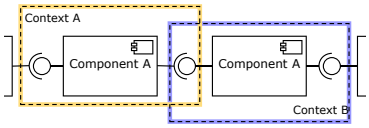
Wiring



- What is connected:
  - Visually (in UML2): interfaces
  - In fact: Roles + *Contexts* → next slides

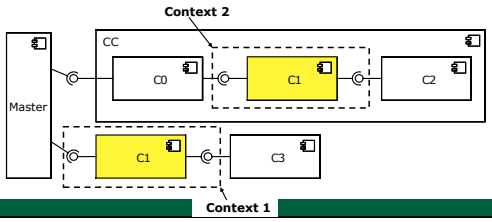
Context (1)

- The same component type can be used multiple times – here: "Component A"

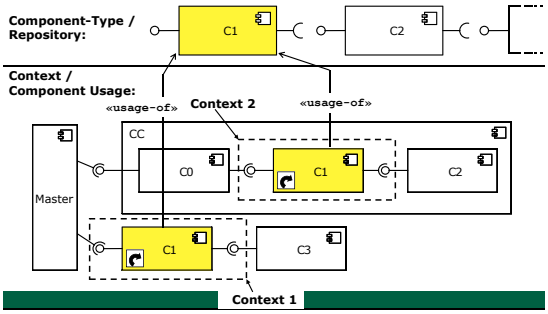


Context (2)

- Structural context (inclusion into a composite component or connection via interfaces to other components)



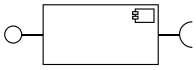
Context (3) - Revealed



Context (4)

- Contexts are often assumed implicitly
- If a component type from a repository is used, usually a context exists by definition

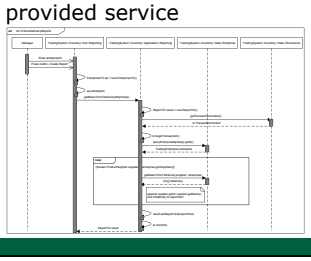
Behavioral Description



- Describes what happens if a provided service is called
- UML provides Sequence diagrams
- SEFF: Service Effect Specification includes component-external effect only

Sequence Diagram

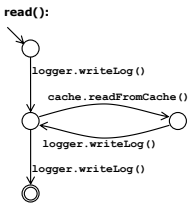
- Can be used to model the effects of a provided service: The entrance call is the provided service



SEFF

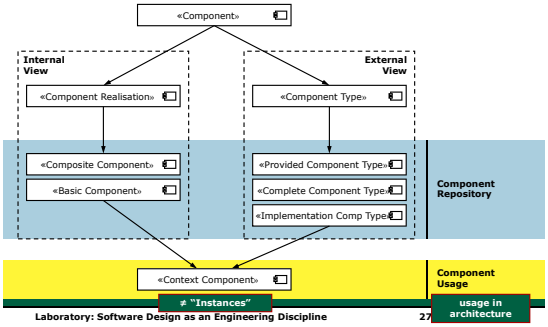
ReaderWriter Component

```
ILogging logger;
ICache cache;
[..]
public string read(Handle h) {
   logger.writeLog("start cache read");
   while(h.hasNext()) {
        cache.readFromCache(h.current());
        logger.writeLog("cache access");
   }
   logger.writeLog("end cache read");
}
```

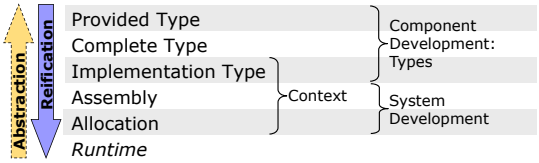


- SEFF as Finite State Machine

Extended view of components



Component Hierarchy



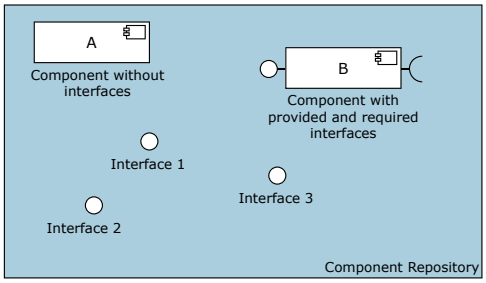
Roles in CBSE

Palladio Process Model and Roles

Roles in the process of component architecture development

- **Overview*:**
- Define component types (Component designer)
- Implement components (Component developer)
- Assemble components (Component assembler)
- Allocate components (Component deployer)
- Deploy whole systems (System deployer)
- * Palladio process view

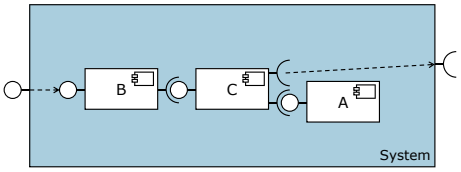
Component Designer



Component based software development (1)

- Define component types
  - No implementation
  - Might include protocol definitions
  - QoS-Attributes of types can be specified
  - Extended component type definitions like "Service Effect Specifications" can be defines
  - (Parameterized) contracts might be used
  - Component types can be defined at different levels (e. g. provided type, complete type, implementation type)
  - →Component types are put into a type-repository

Component Assembler



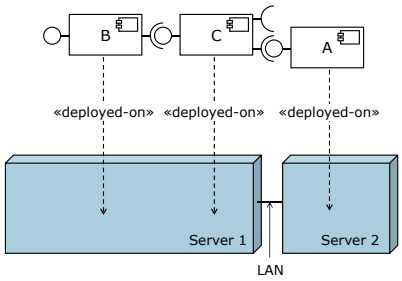
Component based software development (2)

- Assemble components
  - Component types from the repository are inter-connected
  - Assembly and delegation connectors are used
  - If composite components are defined they are put into the component type repository
  - Component architectures are defined
  - →Results in a component architecture "system" (bundle of assembled component) with dedicated system interfaces
  - →Still type level

Component based software development (3)

- Implement components
  - Basic components are implemented
  - (Composite components are realized by other components only, therefore at most factory like structures have to be implement → research)
  - Requirements from the type definition (including prescriptions from extended descriptions like SEFFs) have to be met → what about QoS prescriptions?
  - →Results in implementation instances

Component Deployer



Component based software development (4)

- Allocate components
  - Do the mapping hardware ↔ components
  - *Define* how the hardware environment (especially the distribution) has to look like
  - →Results deployment instances
- Deploy whole systems
  - Make the allocated "system" *run* on a machine
  - →Results in runtime instances

Lessons Learned

- Component hierarchy
- Differences between types and implementations
- Differences at type and implementation levels
- Interface levels
- Understand the idea how to derive SEFFs from source code
- Steps, tasks, and roles of the CBSE development process

Universität Karlsruhe (TH)
Research University ▪ founded 1825

Laboratory: Software Design as an Engineering Discipline

Performance Prediction Tools:
Short Introduction and Demo

Ralf Reussner (reussner@ipd.uka.de)

Chair Software Design and Quality
Institute for Program Structures and Data Organization (IPD)
Faculty of Informatics, Universität Karlsruhe (TH)

Overview

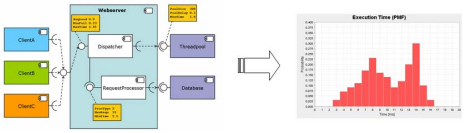
- Performance Prediction
  - Einordnung ins Praktikum
  - Werkzeuge
- Vorbereitung für die nächsten Übungen
  - Eclipse
  - PCM Bench
  - SPE-ED

Revised: Performance Prediction

- Systematischer Entwurf von Software-Systemen
- Einsatz komponentenbasierter Technologien
- Blick auf nicht-funktionale Eigenschaften
  - Im Besonderen: Performanz
  - Verwendung modellgetriebener Vorhersageverfahren
- Verfahren
  - Software Performance Engineering (SPE)
  - Palladio

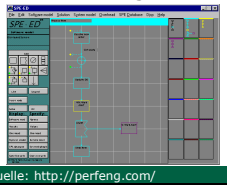
Revised: Werkzeuge

- Vorhersage- und Bewertungsverfahren werden durch Werkzeuge unterstützt



Werkzeug: SPE·ED

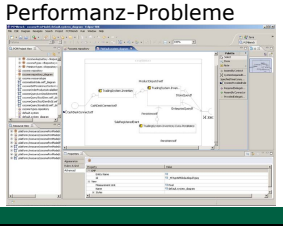
- Autor: Connie U. Smith
- Erstellen und lösen von visuell repräsentierten Performanz-Modellen
- Unterstützung von u. a. Entwurfs-Entscheidungen



Quelle: http://perfeng.com/

Werkzeug: Palladio

- Unterstützung eines gesamten komponentenbasierten Entwurfsprozesses
- Analyseverfahren liefern Hinweise auf Performanz-Probleme



Eclipse

- Für PCM Bench Eclipse mit bestimmter Menge von Plugins benötigt.
- Bereits vorgefertigte Version unter
- http://sdqweb.ipd.uka.de/lehre/SS07-ISE/eclipse.zip
- Aktuelle Versionen der Plugins über Update Site
- http://sdqweb.ipd.uka.de/eclipse/PCM_stud

Palladio Demo

Verwendung SPE-ED

- Herunterladen des Tools von
- http://sdqweb.ipd.uka.de/lehre/SS07-ISE/speed3b.zip
- Im Moment nur eine ältere Version des Tools verfügbar
  - Nach neuer Version wurde angefragt
  - Bis dahin: Systemzeit vor dem Start zurückstellen (Jahr 2003)
  - Kann nach dem Start während der Arbeit bereits wieder korrigiert werden.
- Gesamtes SPE-ED Verzeichnis ins SVN einchecken

SPE-ED Demo

Practical sessions

- Freitags: Praxis (Raum 356)
  - 9:45 bis 11:30 Uhr
  - Weitere Poolraum-Nutzung („exklusiv")
    - Mittwochs von 14:00 bis 17:30 Uhr
    - Freitags von 9:45 - 11:30 Uhr
    - Zugang → später
- Bringen Sie bitte pro Übungsgruppe ein Notebook mit Microsoft Windows mit!
  - Bei den nächsten drei Terminen
  - Übungsgruppen, die diese Möglichkeit nicht haben?

SVN

- Gab es Probleme bei der Verwendung von SVN?
  - Im Anschluss Fragen klären.

Lessons Learned

- How to install and use the PCM Bench
  - Obtain needed Eclipse version
  - Update plugins
  - Create a simple project
- How to use the SPE-ED tool
  - Start the SPE-ED tool (time settings!)
  - Create a software model
  - Solve a model

A.2 SPE-ED Tutorial Slides

The tutorial slides were originally created by Heiko Koziolek for the experiment described in [Koz04] and extended for this experiment.

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Software Performance Engineering (SPE)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

Inhalt nächste Woche

- System Execution Models
  - Queueing Network
- Analyse der Ergebnisse
- Konkurrenzsituationen
- Verteilte Systeme
- Schätzungen
- Komplexeres Beispiel

Motivation

- Performanz
  - wichtige nicht-funktionale Eigenschaft von Software-Systemen
  - umfasst z.B. Antwortzeiten, Durchsatz, Auslastung…
- Häufiger Grund für Performanzprobleme: Architekturmängel
- Nachträgliche Änderungen der Architektur nach erfolgter Implementierung teuer

→ Performanzanalysen bereits beim Design der Architektur

Beispiel für mangelnde Performanz

- Satellitenstart der NASA
  - Ökosystem der Erde untersuchen
  - Projektvolumen EOSDIS: Mehrere Milliarden Dollar
  - Flight Operation Segment Software von Lockheed Martin Corp. hatte inakzeptable Antwortzeiten
  - 8 Monate Verzögerung
  - Kosten unbekannt

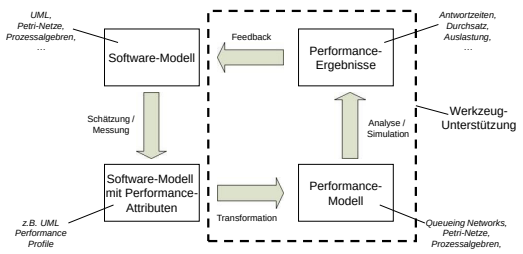
Heather Harreld. NASA delays satellite launch after finding bugs in software program. In Federal Computer Week, 20. April 1998.

Beispiel für mangelnde Performanz

- Online Brokerage Web Sites
  - Nachgeben der Aktienkurse am 27.10.1997
  - Führte zu erhöhter Anfragezahl
  - Web Server waren überlastet
  - Verluste bei Anlegern
  - Klage gegen Betreiber

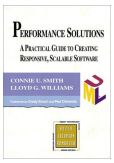
Connie U. Smith. Performance Solutions: A Practical Guide To Creating Responsive, Scalable Software. Addison-Wesley, 2002.

Performanzanalyse von Software-Architekturen



Software Performance Engineering

- 1981: Prägung des Begriffs „Software Performance Engineering“ durch Connie U. Smith
- 1990: Buch „Performance Engineering of Software Systems“
  - erste vollständige Methode zur Performance-Analyse von Software-Systemen während des Entwurfs
- 2002: Buch „Performance Solutions“
  - Integration der UML
  - Anpassungen für verteilte bzw. eingebettete Systeme
  - Tool SPE-ED
  - Performance-Patterns/Antipatterns



SPE Ansatz

- Performanz nur Rechtzeitigkeit
- Ziel: Gutes Antwortverhalten und Skalierbarkeit erreichen
  - Wirkliches und gefühltes Antwortverhalten
- Dazu proaktives Vorgehensmodell
  - Mix aus Modellieren, Messen und anderem
  - Prinzipien, Patterns, Antipatterns
- Hauptsächlich in früher Entwurfsphase
  - Nicht erst warten bis gemessen werden kann
  - Aber auch für bereits bestehende Probleme

Inhalt heute

- Motivation

→Vorgehensmodell

- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Beispiel: Entwurf eines Geldautomaten

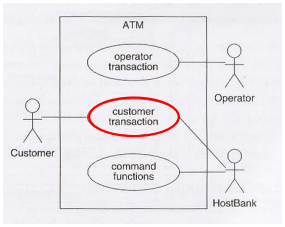
Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Festlegung des Umfangs des SPE-Projekts je nach Signifikanz der Performance
- Erstellung eines detaillierteren Modells bei Performance-kritischeren Anwendungen
- Beispiel Geldautomat: geringes Performance-Risiko, einfaches Modell ausreichend

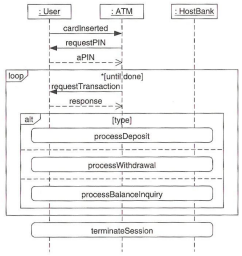
Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells



Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells



Vorgehensmodell

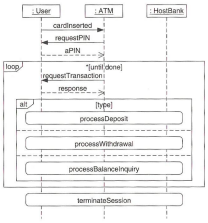
1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Konkrete Vorgaben für z.B. Antwortszeiten, Durchsatz, Ressourcennutzung usw.
- Beispiel Geldautomat: „Die Bedienung darf nicht länger als 30 Sekunden dauern."

Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

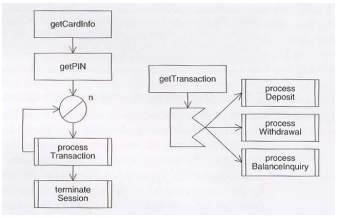
- Transformation der Sequenzdiagramme in Ausführungsgraphen



Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

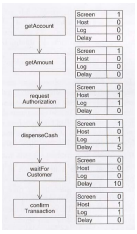
- Transformation der Sequenzdiagramme in Ausführungsgraphen



Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Bestimmung der Anforderungen von Software-Elementen



Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Bestimmung der Anforderungen von Hardware-Elementen

Devices	CPU	Disk	Display	Delay	Net
Quantity	1	1	1	1	1
Service Units	Sec.	Phys. I/O	Screens	Units	Msgs.
Screen	0.001		1		
Host	0.005			3	2
Log	0.001	1			
Delay				1	
Service Time	1	0.02	1	1	0.05

Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Berechnung der Performance-Werte
- Beispiel Geldautomat: „Die Bedienung dauert 29 Sekunden und ist somit knapp innerhalb der Vorgaben"
- bei Nichteinhaltung der Vorgaben:
  - Modifikation der Software
  - Modifikation der Szenarien
  - Überarbeitung der Performance-Ziele

Vorgehensmodell

1. Eingrenzung des Performance-Risikos
2. Identifikation kritischer Use-Cases
3. Modellierung von Performance-Szenarien
4. Festlegung von Performance-Zielen
5. Konstruktion des Performance-Modells
6. Software Resource Requirements
7. Computer Resource Requirements
8. Auswertung des Performance-Modells

- Parallele Aufgaben
  - Verifikation der Modelle
    - Wird das Modell richtig erstellt?
    - Entsprechen die Vorhersagen der Modelle der Performanz des Systems?
  - Validierung der Modelle
    - Wird das richtige Modell erstellt?
    - Stellt das Modell die Ausführungs-charakteristika der Software dar?

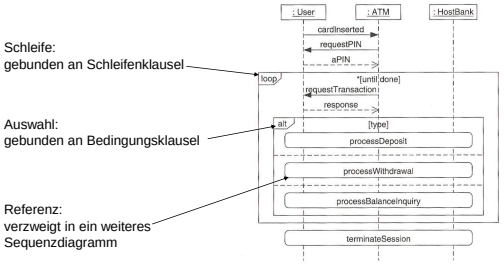
Inhalt heute

- Motivation
- Vorgehensmodell
- →SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

SPE und UML

- UML: Standard-Notation für die Modellierung objektorientierter Systeme
- Basis für die Erstellung des Performance-Modells
- Erweiterungen der UML für Performance-spezifische Informationen (angelehnt an Message Sequence Charts)
  - hierarchische Strukturierung von Sequenzdiagrammen
  - graphische Repräsentation von Schleifen und Auswahlknoten in Sequenzdiagrammen
  - Nebenläufigkeit in Sequenzdiagrammen
- Mit UML 2 ausdrückbar

Beispiel: Sequenzdiagramme



Sequenzdiagramme: Kontrollfluss

- Prozeduraufrufe: Fortsetzung der Ausführung erst nach Beendigung aller verschachtelten Aufrufe
- Nichtprozeduraler Aufruf: nächster Schritt in der Sequenz
- Asynchrone Kommunikationen zwischen zwei Objekten
- return-Anweisung eines Prozeduraufrufs

Erweiterte Features für Sequenzdiagramme:

- Referenz
  - Details in einem weiteren Sequenzdiagramm
- Iteration
  - n-malige Wiederholung des eingeschlossenen Abschnitts
- Alternative
  - Ausführung genau eines Abschnittes gemäß der Bedingungsklausel
- Optionale Abschnitte
  - Ausführung des eingeschlossenen Abschnitts optional gemäß der Bedingungsklausel
- Parallele Abschnitte
  - gleichzeitige Ausführung der eingeschlossenen Abschnitte

Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme

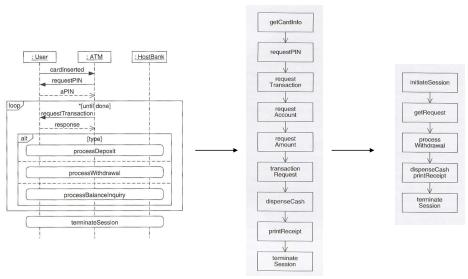
→Software Execution Models

  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

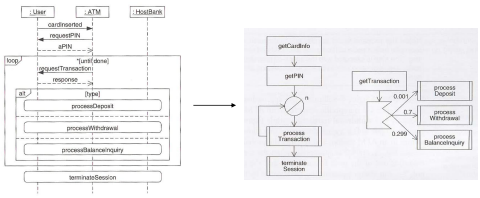
Vom Sequenzdiagramm zum Ausführungsgraph

- Execution Graph
  - Modell zur vereinfachten Aufnahme von Performance-Informationen
  - Ähnlichkeit mit UML-Aktivitätsdiagrammen
- Umwandlung jedes Aufrufes im Sequenzdiagramm in einen Knoten
- Zusammenfassung verwandter bzw. Performance-unkritischer Aufrufe in einen einzigen Knoten
- Ergänzung von Ausführungshäufigkeiten / -wahrscheinlichkeiten

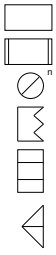
Beispiel: Sequenzdiagramm -> Ausführungsgraph



Beispiel: Sequenzdiagramm -> Ausführungsgraph



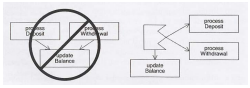
Ausführungsgraphen



- Basisknoten
  - Funktionale Komponente
- Erweiterter Knoten
  - Verfeinerte Funktion in einem Subgraphen
- Wiederholungsknoten
  - Schleife: n-malige Wiederholung einer Reihe von Knoten
- Entscheidungsknoten
  - Ausführung geknüpft an Bedingung
  - versehen mit Wahrscheinlichkeit
- Parallelknoten
  - parallele Ausführung von Knoten
  - Fortsetzung des Kontrollflusses erst nach Berechnung aller Knoten
- Spaltungsknoten
  - parallele Ausführung von Knoten
  - Fortsetzung des Kontrollflusses auch wenn nicht alle Knoten berechnet wurden
- Knoten zur Synchronisation später

Ausführungsgraphen: Nicht erlaubte Konstrukte

- mehrere Anfangsknoten für einen Graphen



- Schleifen ohne Wiederholungsknoten

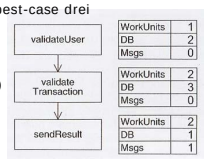
Eingabedaten für SPE

1. Performance-kritische Szenarios
   - modelliert als Sequenzdiagramme, Ausführungsgraphen
2. Performance-Ziele
   - konkrete Vorgaben, die zu erfüllen sind
3. Systemumgebung
   - Hardware-Konfiguration, Betriebssystem, Middleware
4. Software Resource Requirements
   - Berechnungsanforderung für jeden Rechenschritt
5. Computer Resource Requirements
   - Abbildung der Software Resource Requirements auf die Ausführungsumgebung

in der Übung vorgegeben

Software Resource Requirements

- Identifikation der Software Ressourcen
  - Beispiel: Berechnungseinheiten, Datenbankzugriffe, Netzwerknachrichten
- Schätzung der Werte für die Software-Ressourcen
  - **best- und worst-case**, später detaillierter
  - Beispiel: Operation x braucht im best-case drei Datenbankzugriffe
- Angabe der CPU-Belastung
  - Beispiel: WorkUnits über einer Skala von 1 (niedrig) bis 5 (hoch)
- Beispiel: Autorisierung einer Transaktion mittels einer Datenbank
  - hier: Spezifikation der best-case Werte

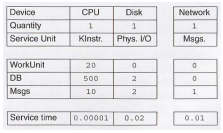


Software Resource Requirements

Software Resource Type	Einheit
CPU usage	Work Units, Anzahl der Instruktionen oder Zeit
SQL	Anzahl und Typ (read, write, update, ...)
File I/O	Anzahl der logischen oder physischen I/Os
Messages	Größe (z.B. in KByte) und Typ (LAN, Internet)
Log-Files	Anzahl der Log-Events
Middleware-calls	Anzahl und Typ (connectionOpen, queueGet, requestSend, ...)
Calls to different processes, threads or processors	Anzahl, Typ und Ziel des Aufrufs
Delay for remote processing	Zeit eines Requests

Computer Resource Requirements

- Abbildung der Software Anforderungen auf Hardware-Elemente
- Typ, Anzahl und Einheit der relevanten Hardware-Elemente im System
- Spezifikation, wie hoch die Software Requirements auf den Computer Ressource Requirements sind
- Dauer einer Einheit für jede Ressource (z.B. durch Messungen)

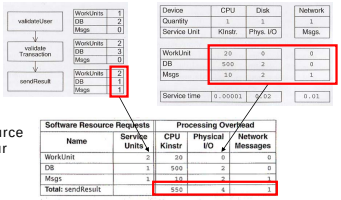


Computer Resource Requirements

Computer Resource Type	Einheit
CPU	Zeit, Anzahl der Instruktionen
Disks, I/O	Anzahl der Operationen
Benutzer	Zeit
Netzwerk	Anzahl der Nachrichten
Verzögerung (ohne Queue, z.B. Bildschirm, Tastatur...)	Zeit
Box (mit Queue, z.B. Drucker)	Zeit

Computer Resource Requirements: Berechnung der Antwortzeit

Schritt 1: Berechnung der Computer Resource Requirements für einen Knoten (sendResult)

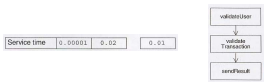


Computer Resource Requirements: Berechnung der Antwortzeit

Schritt 2: Berechnung der Computer Resource Requirements für alle Knoten eines Graphen

Processing Step	CPU Kinstr	Physical I/O	Network Messages
validateUser	1,020	4	0
validateTransaction	1,540	6	0
sendResult	550	4	1
Total: authorizeTransaction	3,110	14	1

Schritt 3: Berechnung der best-case Antwortzeit (ggf. Einbeziehung von Iterationen und Wahrscheinlichkeiten bei Auswahlknoten)



(3110 * 0,00001) + (14 * 0,02) + (1 * 0,01)
=
0,3211 Sekunden

Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs

→Beispiel

- Performance-orientiertes Design
- SPE-ED Hinweise

Beispiel

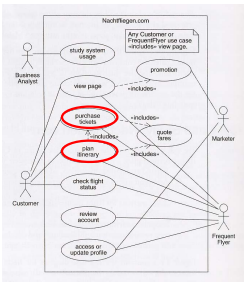
- Web-Auftritt einer Fluggesellschaft
- Planung von Flugreisen im Web
- Bestellen von Tickets
- Schalten von Werbung
- Vielflieger-Konten und –Angebote

- Verwendet Altsysteme der Fluggesellschaft

1. Eingrenzen des Performance-Risikos

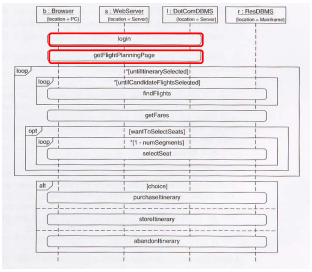
- mittleres Performance-Risiko
- bei schlechter Performance: keine Benutzung der Website -> Umsatzeinbußen

2. Performance-kritische Use-Cases



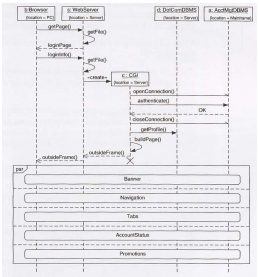
3. Modellierung von Szenarien

planItinerary



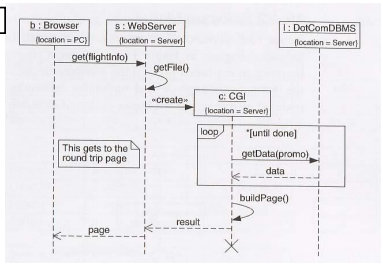
3. Modellierung von Szenarien

login



3. Modellierung von Szenarien

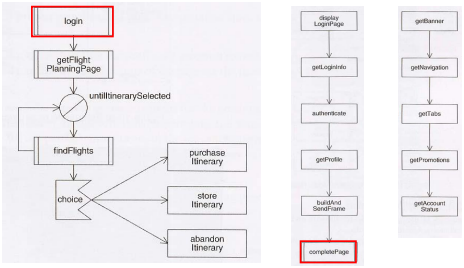
getFlightPlanningPage



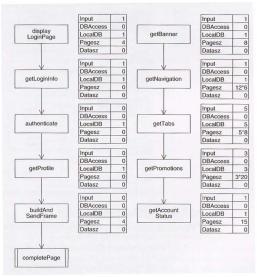
4. Festlegung von Performance-Zielen

- Maximale Login-Zeit: 8 Sekunden
- Ansonsten zunächst keine Vorgaben

5. Konstruktion eines Performance-Models



6. Software Resource Requirements

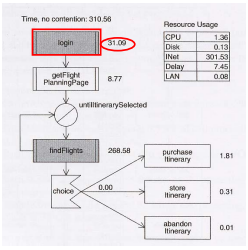


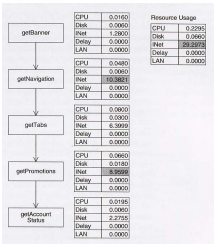
- Input: Größe der Eingabenachricht (KByte)
- DBAccess: Anzahl der Zugriffe auf die Mainframe-Datenbank
- LocalDB: Anzahl der Zugriffe auf das DotComDBMS
- Pagesz: Größe der Seite, die dem Benutzer gezeigt wird (KByte)
- Datasz: Größe der Daten, die aus dem Mainframe empfangen werden (KByte)

7. System Resource Requirements

Devices	CPU	Disk	INet	Delay	LAN
Quantity	6	3	1	1	1
Service Units	sec	Phys I/O	KBytes	sec	Msgs
Input	0.002		1		
DBAccess	0.0005			0.25	1
LocalDB	0.01	2			
Pagesz	0.0005		1		
Datasz	0.0005		1		1
Service Time	1	0.003	.14222	1	0.000164

8. Auswertung des Performance-Modells

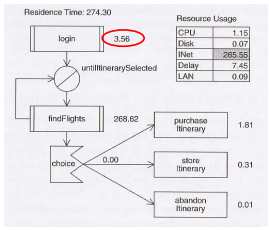




Änderung des Entwurfs

- Beispiele:
  - kleinere Webseiten verschicken
  - Verwendung von Tabellen anstatt von Frames um mehrfache Requests zu umgehen
  - Möglichkeit zur Festlegung einer Startseite durch den User zur schnelleren Navigation
  - Verwendung von weniger aufwendigen Grafiken
  - ...

Änderung des Entwurfs



Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel

→Performance-orientiertes Design

- SPE-ED Hinweise

Performanzprinzipien

Performance Objectives Principle	Define specific, quantitative, measurable performance objectives for performance scenarios.
Instrumenting Principle	Instrument systems as you build them to enable measurement and analysis of workload scenarios, resource requirements, and performance objective compliance.
Centering Principle	Identify the dominant workload functions and minimize their processing.
Fixing-Point Principle	For responsiveness, fixing should establish connections at the earliest feasible point in time, such that retaining the connection is cost-effective.
Locality Principle	Create actions, functions, and results that are close to physical computer resources.
Processing Versus Frequency Principle	Minimize the product of processing times frequency.
Shared Resources Principle	Share resources when possible. When exclusive access is required, minimize the sum of the holding time plus the scheduling time.
Parallel Processing Principle	Execute processing in parallel (only) when the processing speedup offsets communication overhead and resource contention delays.
Spread-the-Load Principle	Spread the load when possible by processing conflicting loads at different times or in different places.

Performanz-Patterns

Pattern	Description	Principle(s)
Fast Path	Identify dominant workload functions and streamline the processing to do only what is necessary	Centering
First Things First	Focus on the relative importance of processing tasks to ensure that the least important tasks will be the ones omitted if everything cannot be completed within the time available	Centering
Coupling	Match the interface to objects with their most frequent uses	Centering Locality Processing Versus Frequency
Batching	Combine requests into batches so the overhead processing is executed once for the entire batch instead of for each individual item	Processing Versus Frequency
Alternate Routes	Spread the demand for high-usage objects spatially, that is, to different objects or locations	Spread-the-Load
Flex Time	Spread the demand for high-usage objects temporally, that is, to different periods of time	Spread-the-Load
Slender Cyclic Functions	Minimize the amount of work that must execute at regular intervals	Centering

Performanz-Antipatterns

Antipattern	Problem	Solution	Principle Violated
"god" Class	Occurs when a single class either 1) performs all of the work of an application or 2) holds all of the application's data. Either manifestation results in excessive message traffic that can degrade performance.	Refactor the design to distribute intelligence uniformly over the application's top-level classes, and to keep related data and behavior together.	Locality Processing Versus Frequency
Excessive Dynamic Allocation	Occurs when an application unnecessarily creates and destroys large numbers of objects during its execution. The overhead required to create and destroy these objects has a negative impact on performance.	1) "Recycle" objects (via an object "pool") rather than creating new ones each time they are needed. 2) Use the Flyweight pattern to eliminate the need to create new objects.	Fixing-Point Locality Processing Versus Frequency

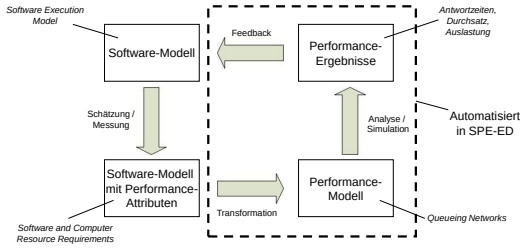
Circuitous Treasure Hunt	Occurs when an object must look in several places to find the information that it needs. If a large amount of processing is required for each "look," performance will suffer.	Refactor the design to provide alternative access paths that do not require a Circuitous Treasure Hunt (or to reduce the cost of each "look").	Locality Fixing-Point
One-Lane Bridge	Occurs at a point in execution where only one, or a few, processes may continue to execute concurrently (e.g., when accessing a database). Other processes are delayed while they wait for their turn.	To alleviate the congestion, use the Shared Resources Principle to minimize conflicts.	Shared Resources Spread-the-Load
Traffic Jam	Occurs when one problem causes a backlog of jobs that produces wide variability in response time which persists long after the problem has disappeared.	Begin by eliminating the original cause of the backlog. If this is not possible, provide sufficient processing power to handle the worst-case load.	Shared Resources Spread-the-Load

Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design

→SPE-ED Hinweise

SPE-ED



SPE-ED

- Werkzeug zur Unterstützung des SPE Prozesses
- Modellierung von
  - Software Execution Models
  - System Execution Models
- Analyse der Modelle
- Anzeige der Analyseergebnisse in Modellen

Struktur

- Verschiedene Sichten
  - Software Model
  - System Model
  - Overhead
  - Results
- Projekte
  - Begleiten einen Entwurf
  - Szenarien für verschiedene Anwendungsfälle / Prozesse
  - Entwurfsalternativen über verschiedene Projekte

Templates

- Um verfügbare Ressourcen zu modellieren
- Für jedes Szenario
- Facility Template
  - Enthält Geräte eines Rechners
  - Performanzwerte
- Software Resource Template
- Kombination: Overhead Matrix

Devices	CPU	Disk	INet	Delay	LAN
Quantity	6	3	1	1	1
Service Units	sec	Phys I/O	KBytes	sec	Msgs

Input	0.002		1		
DBAccess	0.0005			0.25	1
LocalDB	0.01	2			
Pagesz	0.0005		1		
Datasz	0.0005		1		1
Service Time	1	0.003	.14222	1	0.000164

Tipps zu SPE-ED

- In der System Model Ansicht nicht Values klicken (Absturz)

Inhalt heute

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Software Performance Engineering (SPE)

Zweite Sitzung

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Organisatorisches

- Übungszettel heute Nacht
- Einteilung Übungsgruppen neu
- Freitag Übung 9:45 Uhr

Inhalt letzte Woche

- Motivation
- Vorgehensmodell
- SPE und UML
  - Erweiterte Sequenzdiagramme
- Software Execution Models
  - Execution Graphs
- Beispiel
- Performance-orientiertes Design
- SPE-ED Hinweise

Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED
- Daten und Schätzen
- Beispiel ICAD

Konkurrenzsituationen

- Software Execution Model:
  - zum schnellen Feedback von Performance-Problemen in frühen Entwurfsphasen
  - keine Berücksichtigung von anderen Arbeitslasten oder mehreren Benutzern
  - Vernachlässigung der Verzögerungen, die bei konkurrierender Ressourcennutzung auftreten
- Aber: Nicht nur ein Prozess verwendet die Ressourcen
- Verschiedene Konkurrenzsituationen
  - Mehrere Benutzer des Systems
  - Andere Systeme, die die Ressourcen mitbenutzen
  - Ein Szenario enthält parallel auszuführende Teile

Umsetzungen SPE-ED

- No Contention
  - Einzelner Benutzer in einem einzelnen Szenario
  - Analytisch
- Contention
  - Mehrere Benutzer in einem einzelnen Szenario
  - Analytisch mit Warteschlangennetzen
- Simulation
  - Mehrere Benutzer in mehreren Szenarien
  - Nicht analytisch

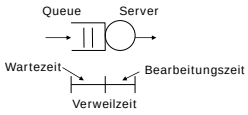
Inhalt heute

- Konkurrenzsituationen
  - →System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED
- Daten und Schätzen
- Beispiel ICAD

System Execution Models

- Software Execution Model:
  - zum schnellen Feedback von Performance-Problemen in frühen Entwurfsphasen
  - keine Berücksichtigung von anderen Arbeitslasten oder mehreren Benutzern
  - Vernachlässigung der Verzögerungen, die bei konkurrierender Ressourcennutzung auftreten
- System Execution Model:
  - Konstruktion erst nach Lösung aller Performance-Probleme im Software Execution Model
  - Modellierung von konkurrierenden Ressourcenzugriffen mittels Warteschlangennetzen (engl.: Queueing Networks)
  - zur Identifikation von Flaschenhälsen in der System-Architektur
  - automatisierte Berechnung durch Tools

Warteschlangen



- Performance Metriken:
  - Verweilzeit
  - Auslastung
  - Durchsatz
  - Queue-Länge

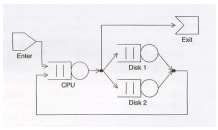
Jeweils im Durchschnitt für einen Server

Warteschlangen

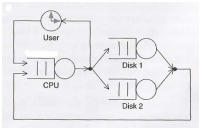
- Annahme: Job Flow Balance
  - Rate der fertiggestellten Jobs (Durchsatz) = Ankunftsrate
- Beispiel:
  - Vorgabe:
    - Ankunftsrate λ = 0,4 jobs/sec
    - Mittlere Bearbeitungszeit S = 2 sec
  - Berechnung:
    - Durchsatz, X = λ — 0,4 jobs/sec (job flow balance)
    - Auslastung, U = X * S — 0,4/sec * 2 sec = 0,8
    - Verweilzeit, RT = S / (1-U) — 2 sec / (1-0,8) = 10 sec
    - Queue-Länge, N = X * RT — 0,4/sec * 10 sec = 4 jobs (Little's Formel)

Warteschlangennetze

- offenes QN:
  - Eingang/Ausgang in das Netzwerk
  - Anzahl der Jobs variiert mit der Zeit



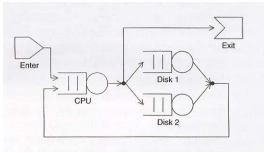
- geschlossenes QN:
  - keine externen Ankünfte im Netzwerk
  - Zirkulation fester Anzahl von Jobs



Offene Warteschlangen-netze: Berechnungen

- Vorgabe:
  - λ Ankunftsrate
  - V_i Anzahl der Besuche (Visits) eines Gerätes
  - S_i Mittlere Bearbeitungszeit (Service Time) eines Gerätes
- Berechnung:
  - Systemdurchsatz X_0: $X_0 = \lambda$
  - Durchsatz von Gerät i: $X_i = X_0 * V_i$
  - Auslastung von Gerät i: $U_i = X_i * S_i$
  - Verweilzeit von Gerät i: $RT_i = S_i / (1-U_i)$
  - Länge der Queue Gerät i: $N_i = X_i * RT_i$
  - Länge der Systemqueue: $N = \Sigma N_i$
  - Systemantwortzeit: $RT = N / X_0$

Offene Warteschlangen-netze: Beispiel

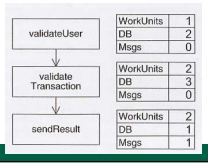


- Systemankunftsrate:
  - λ = 5 jobs / sec = X_0
- Anzahl der Besuche V:
  - CPU 5
  - Disk1 3
  - Disk2 1
- Mittlere Bearbeitungszeit S:
  - CPU 0,01
  - Disk1 0,03
  - Disk2 0,02

Metrik	CPU	Disk1	Disk2
1. Durchsatz $X_i = X_0 * V_i$	25	15	5
2. Bearbeitungszeit S_i (Vorgabe)	0,01	0,03	0,02
3. Auslastung $U_i = X_i * S_i$	0,25	0,45	0,10
4. Verweilzeit $RT_i = S_i / (1-U_i)$	0,013	0,055	0,022
5. Queue Länge $N_i = X_i * RT_i$	0,325	0,825	0,111
Gesamte Jobs im System = 0,325 + 0,825 + 0,111 = 1,261			
Systemantwortzeit: RT = N / X_0 1,261 / 5 = 0,252 sec			

Beispiel

- Beispiel Geldautomat
- Diesmal Bankrechner betrachtet
- Soll Transaktion prüfen und ausführen
- Informationen aus Software Execution Graph:



Device	CPU	Disk	Network
Quantity	1	1	1
Service Unit	Kinstr.	Phys. I/O	Msgs.
WorkUnit	20	0	0
DB	500	2	0
Msgs	10	2	1
Service time	0.00001	0.02	0.01

System Execution Model

- Eingaben für das Warteschlangennetz
  - Systemankunftsrate (ermittelt in einem Performance Walkthrough)
  - Anzahl der Besuche an einem Gerät (berechnet mit dem Software Execution Model)

Processing Step	CPU Kinstr	Physical I/O	Network Messages
validateUser	1,020	4	0
validateTransaction	1,540	6	0
sendResult	550	4	1
Total: authorizeTransaction	3,110	14	1

  - Mittlere Bearbeitungszeiten (ebenfalls im Software Execution Model festgelegt)

Service time	0.00001	0.02	0.01

- Berechnung der Ausgaben (Systemantwortzeit, Queue-Längen usw.) mit Tool

Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze

  →System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED
- Daten und Schätzen
- Beispiel ICAD

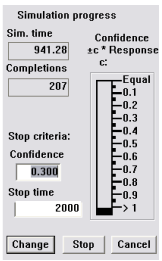
Simulation

- Mehrere Szenarien gleichzeitig betrachten
- Auch: System Model Solution
- Simulation mit CSIM http://www.csim.com/

- Warteschlangennetz für jede Facility
  - Mit Anforderungen aus No Contention Lösung
  - Prioritäten für Szenarien
  - Ankunftszeiten bzw. Denkzeit exponentialverteilt

Simulation

- Durchlauf der Szenarien
- Abbruchbedingungen
  - Zeitbegrenzung
  - Bestimmtes Konfidenzintervall erreicht
    - Je größer Wert, desto ungenauer



Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen

→Analyse der Ergebnisse

- Weitere Konstrukte in SPE-ED
- Daten und Schätzen
- Beispiel ICAD

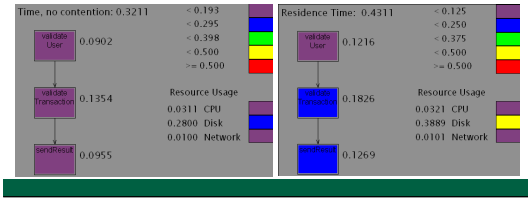
Analyse der Ergebnisse

- Ergebnisse interpretieren je nach Konkurrenzsituation
  - No Contention
  - Contention
  - Simulation

- Darauf basierend für Entwurfsalternative entscheiden
  - Oftmals nach bester Antwortzeit
  - Aber auch andere Kriterien

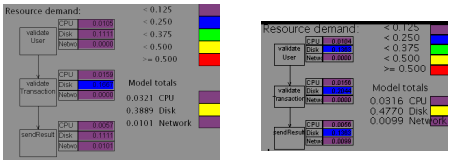
Ergebnisse: Verweilzeit

- Residence Time
  - Gesamtverweilzeit des Szenarios
  - Verweilzeit pro Knoten
  - Gesamtverweilzeit pro Gerät
- Hier: No Contention und Contention, 1 User/Sek



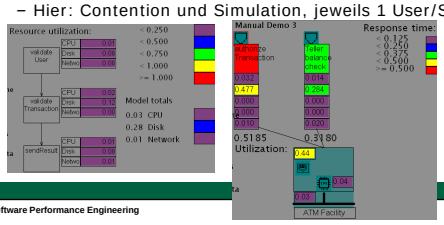
Ergebnisse: Last

- Resource Demand
  - Verweilzeit pro Knoten pro Gerät
  - Gesamtverweilzeit pro Gerät
- Hier: Contention und Simulation, je 1 User/Sek



Ergebnisse: Auslastung

- Utilization
  - Auslastung der Geräte pro Knoten
  - Gesamtauslastung der Geräte
  - Bei Auslastung >1 kann Gerät Anfragen nicht bedienen
  - Hier: Contention und Simulation, jeweils 1 User/Sek



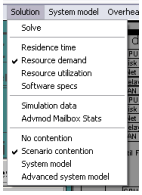
Interpretation

- Vergleich mit Performanzziel
- Bewertung der besten Alternative hinsichtlich der Antwortzeit ist klar

- Aber Bewertung der besten Alternative?
- Auch andere Faktoren, ggf. Vorteile langsamerer Alternativen
  - Erweiterbarkeit
  - Skalierungsfragen
  - ...

Tipps SPE-ED

- Tipp: Verschiedene Ergebnisansichten über das Menü auswählen
- Zwischenergebnisse als andere Projekte speichern
  - Kopie der entsprechenden Dateien



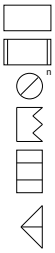
Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse

→Weitere Konstrukte in SPE-ED

- Daten und Schätzen
- Beispiel ICAD

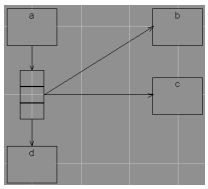
Ausführungsgraphen

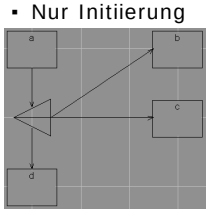


- Basisknoten
  - Funktionale Komponente
- Erweiterter Knoten
  - Verfeinerte Funktion in einem Subgraphen
- Wiederholungsknoten
  - Schleife: n-malige Wiederholung einer Reihe von Knoten
- Entscheidungsknoten
  - Ausführung geknüpft an Bedingung
  - versehen mit Wahrscheinlichkeit
- Parallelknoten
  - parallele Ausführung von Knoten
  - Fortsetzung des Kontrollflusses erst nach Berechnung aller Knoten
- Spaltungsknoten
  - parallele Ausführung von Knoten
  - Fortsetzung des Kontrollflusses auch wenn nicht alle Knoten berechnet wurden
- Knoten zur Synchronisation

Parallelität im Ausführungsgraphen

- Pardo (Parallel Do)
- Warten auf alle
- Split
- Nur Initiierung

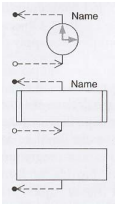




Jeweils mit Wahrscheinlichkeiten

Ausführungsgraphen: Synchronisation

Aufrufender Prozess:

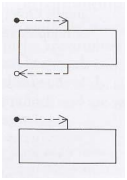


Synchroner Aufruf: der Aufrufer wartet auf eine Antwort

Verzögerter synchroner Aufruf: Weiterrechnen, danach Warten auf Antwort

Asynchroner Aufruf: keine Antwort

Aufgerufener Prozess:



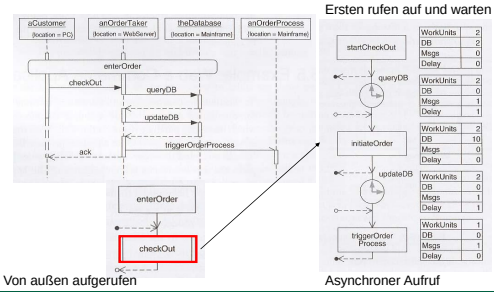
Berechnung, dann Antwort

keine Antwort

Lesen: Aufruf bzw. , Antwort bzw.

Ausführungsgraphen: Synchronisation - Beispiel

Ersten rufen auf und warten



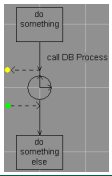
Von außen aufgerufen

Asynchroner Aufruf

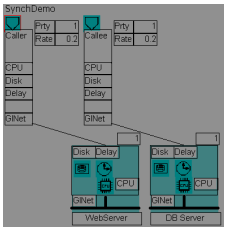
Synchronisation zweites Beispiel

- Aufruf einer Datenbank auf einem zweiten Server
- 12 Benutzer pro Minute

Szenario Caller ruft DB Prozess auf



Szenario Callee enthält Subgraphen DB Process



Szenarien auf unterschiedlichen Facilities

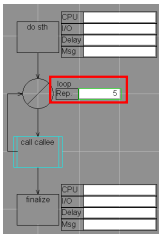
Zuordnung der Knoten

- Synchronisationsknoten müssen implizit einander zugewiesen werden
- Werten Sie den aufgerufenen Knoten aus (mit den benötigten Konkurrenzsituationen)
- Geben Sie beim aufrufenden Knoten die berechnete Verweilzeit als Delay an
- Bedenken Sie, die Ankunftsrate bzw. Anzahl der Benutzer für das aufgerufene Szenario anzupassen, wenn der Aufruf in einer Schleife stattfindet.

Zuordnung der Knoten

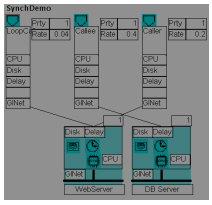
- Beispiel für veränderte Ankunftsraten:

LoopCaller Szenario



Subgraph call callee

Betrachte Ankunftsrate AR(Callee) = AR(LoopCaller) * 5 + AR(Caller)



Konstrukte

- Nicht alle Konstrukte in Übungen
- Trotzdem alle Konstrukte kennen
  - Knapp wie auf den Folien
- Für Hintergrundwissen in Handbuch schauen

Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED

→Daten und Schätzen

- Beispiel ICAD

Eingabedaten für SPE

1. Performance-kritische Szenarios
   - modelliert als Sequenzdiagramme, Ausführungsgraphen
2. Performance-Ziele
   - konkrete Vorgaben, die zu erfüllen sind
3. Systemumgebung
   - Hardware-Konfiguration, Betriebssystem, Middleware
4. Software Resource Requirements
   - Berechnungsanforderung für jeden Rechenschritt

   in der Übung vorgegeben
5. Computer Resource Requirements
   - Abbildung der Software Resource Requirements auf die Ausführungsumgebung

Woher Daten nehmen?

- Typischerweise zu frühem Entwurfszeitpunkt
- Entwurf noch nicht vollständig ausgearbeitet
  - Alternativen für Architekturen
  - Andere mögliche Entwurfsalternativen
- Noch keine (vollständige) Implementierung vorhanden
- Ggf. ähnliche Systeme bekannt
- Rest: Schätzungen

Benötigte Daten

- Welches sind performanzkritische Szenarien?
  - 80-20 Regel
  - Modelle und Benutzungsprofil erstellen
  - Systemarchitekt, Marketing, Benutzer
-
- Was sind die Performanzziele?
  - Feste Vorgaben
  - Ggf. bestimmte Einschränkungen
    - Ressourcennutzung einschränken weil bekannt
    - Auslastung einschränken bei Echtzeitsystemen
  - Systemarchitekt, Marketing, Performance Engineer

Performanzziele

- Performanzziele nicht unbedingt nur einzelne Antwortzeit
  - Maximal 20 Millisekunden
  - Im Mittel 15 ms, höchstens 20
  - Jeweils nur mit gewisser Konfidenz
- Benötigt sind Analysen für:
  - Best-Case,
  - Average-Case,
  - Worst-Case
- Dementsprechende weitere Daten und Schätzungen

Benötigte Daten

- Welche Eigenschaften hat die Systemumgebung?
  - Hardware und Netzwerkkonfigurationen
  - Betriebssysteme, Middleware, …
  - Beteiligte Datenbanken
  - Konkurrierende Software
- System-Deployer

Benötigte Daten

- Software Resource Requirements
  - Anforderungen der einzelnen Knoten
- Verschiedene Typen von Anforderungen
  - Festplatten und Netzwerkzugriffe noch gut zu schätzen
  - Verhalten von Datenbanken
  - Berechnungsaufwand muss abgeschätzt werden
    - Realistische Skala finden
- Später ggf. mehr Typen hinzufügen
- Entwickler, Performance Engineer

Benötigte Daten

- Computer Resource Requirements
  - Abbildung der Software Resource Requirements auf Hardware
  - Abhängig von Ausführungsumgebung
  - Abstraktion ermöglichen
    - 1 DB -> 50.000 CPU Instr + 1 File I/O
  - Performance Engineer, Kapazitätsplaner

Beispielwerte

- Festplatte:
  - 20-30 MB pro Sekunde
  - 5-10 ms Latenz
- Netzwerkverbindung
  - Durchsatz sehr unterschiedlich
    - 128 KBit/s; 1GBit/s; DSL 1000
  - Latenzzeiten von Standorten abhängig
    - Lokal < 1ms, Karlsruhe -> Oldenburg ca. 250 ms, Karlsruhe -> Neuseeland ca. 400 ms

Beispiel

- Aufteilung Netzwerk in zwei Software Resource Requirements
  - Anzahl Nachrichten
    - Damit Latenzzeit berechnen
  - Größe der Nachrichten
    - Übertragungsdauer vervollständigen

Device:	Disk
# I/O	0.001
FileSize I/O	25
Service Time	1

Device	Network
# Messages	0.4
MessageSize	128
Service Time	1

- Evtl. eines weglassen wenn keinen Einfluss

Beispielwerte

- CPU schwer abzuschätzen
  - Feingranulare Instruktionen
  - Instruktionen nicht 1:1 auf Zyklen abzubilden
    - Beispielsweise 1 Instruktion in 0,8 Zyklen
  - Befehlssatz von CPU abhängig
- Abstraktion durch Work Units
- Abbildung auf Computer Resource Requirements finden
  - Messungen ähnlicher Systeme
  - Grobe Schätzungen
  - (Vorgabe)
- Berechnungsaufwand in Work Units einschätzen

Computer Resource Requirements

- In letzter Übung vorgegeben, wie anzugeben
- Diesmal alle Rollen der Performanzanalyse ausfüllen
  - Selbst angeben wie die Matrix spezifiziert wird.
  - In Übung
- Beispiel CAD System

Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED
- Daten und Schätzen

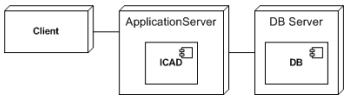
→Beispiel ICAD

Beispielsystem

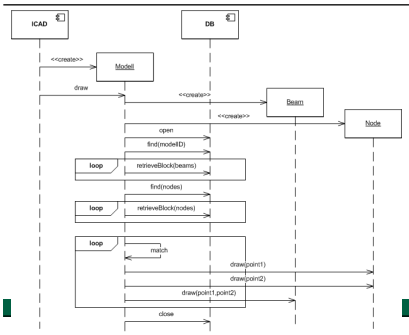
- Interaktives CAD System ICAD
- Ingenieure zeichnen Modelle
- Modelle sind in DB abgelegt
- System validiert und analysiert Modelle
- Zeichnung besteht aus Knoten und Formen
- Koordinaten im dreidimensionalen Raum
- Anwendungsfälle Zeichnen und Bewerten
- Im Mittel alle 5 Sekunden eine Anfrage
- Typisches Modell
  - 1000-3000 Balken
  - Mittelwert ca. 1500 Balken
- Performanzziel: Modell normalerweise in 5 s anzeigen, nicht länger als 10 s

ICAD Szenario

- Jeweils Kenndaten für
  - CPU
  - Disk
  - Netzwerk
- AppServer auch
  - Delay (Screen)

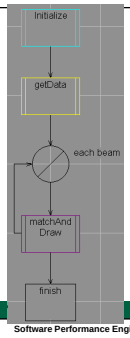


ICAD Szenario



- Block je 100 Balken oder Knoten

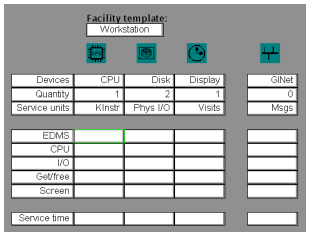
Software Execution Graph



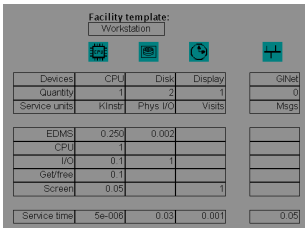
- Software Resource Requirements angeben
- Variable Anzahl Balken
  - Mittelwert 1500 Balken
  - Schlechtester Fall 3000 Balken
- Einfluss auf
  - Anzahl Schleifendurchläufe
  - Ladezeiten aus DB
  - Aufwand Matching
- Zwei Modelle

Computer Resource Requirements erstellen

- Angaben für Abbildungen aus Vorgaben
- Hier:
- Die CPU ist 5GHz schnell → Service Time = 5e-006
- Ein Zugriff auf die EDBMS benötigt ca. 250 KInstr CPU und 2 Festplatten-zugriffe
- ...

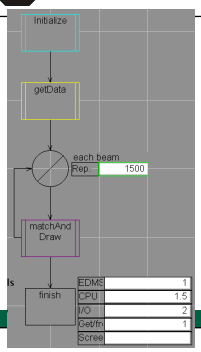


Computer Resource Requirements



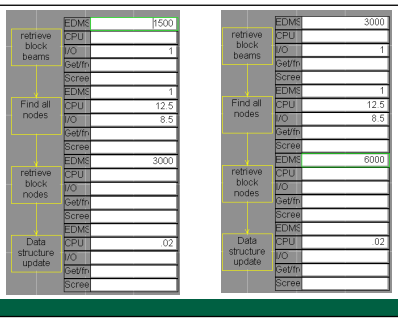
- Weitere Daten eingeben nach Vorgaben

Schleifendurchläufe

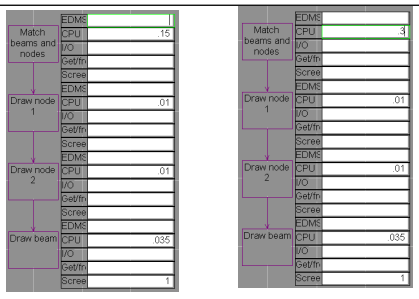


- Mittel 1500
- Maximal 3000
- Zwei Projekte

Ladezeiten

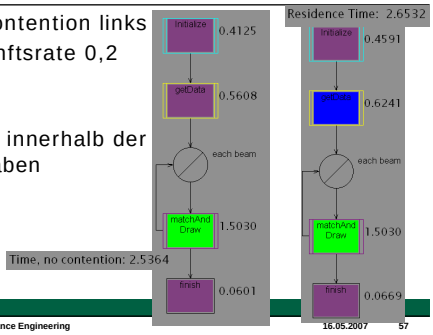


Aufwand Zusammenführung



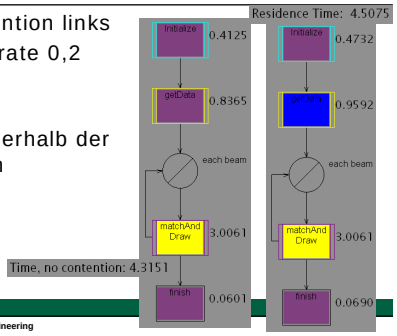
Analysen Mittelwert

- No Contention links
- Ankunftsrate 0,2 links
- Beide innerhalb der Vorgaben



Analysen schlechter Fall

- No Contention links
- Ankunftsrate 0,2 links
- Beide innerhalb der Vorgaben



Warum ist alles so komplex?

- In anderen Ingenieurwissenschaften gibt es doch auch Vorhersagen?
- Aber Software ist anders
  - Halteproblem
  - Daher Hauptproblem: Richtige Abstraktionsebene finden
  - Lösbar, aber noch aussagekräftig

Inhalt heute

- Konkurrenzsituationen
  - System Execution Models
    - Warteschlangennetze
  - System Models
    - Simulationen
- Analyse der Ergebnisse
- Weitere Konstrukte in SPE-ED
- Daten und Schätzen
- Beispiel ICAD

A.3 Palladio Tutorial Slides

The tutorial slides were created by Heiko Koziolek and Steffen Becker for this experiment.

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model (PCM)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
3. Stochastical Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

Lecture 1 / Lecture 2 / Lecture 3

Outline

4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation
6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Solver, Result Interpretation
8. Comprehensive Case Study
9. Outlook

Lecture 4 / Lecture 5

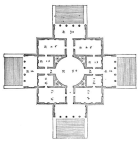
Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
3. Stochastical Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

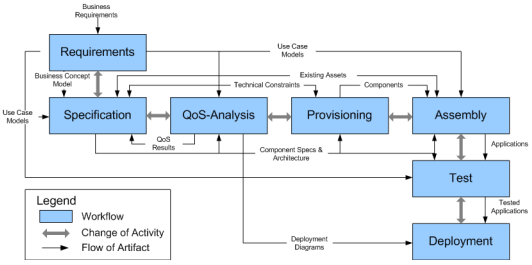
Lecture 1 / Lecture 2 / Lecture 3

Palladio Component Model

- Developed at Uni Oldenburg, Uni Karlsruhe since 2003
- Domain-specific Modelling Language
- Targeted at
  - Performance Prediction for Component-based Software Architectures
  - Business Information Systems
- Extensive Metamodel in EMF/Ecore
- Named after famous Renaissance Architect

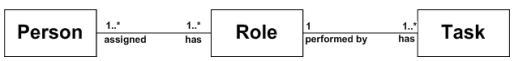


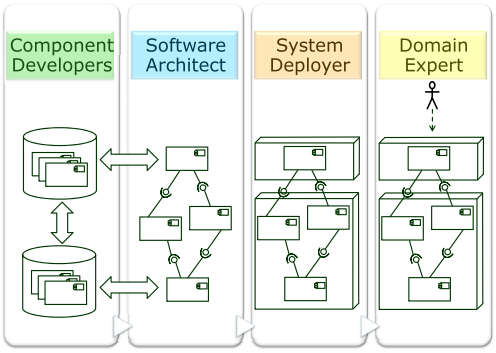
CBSE Development Process



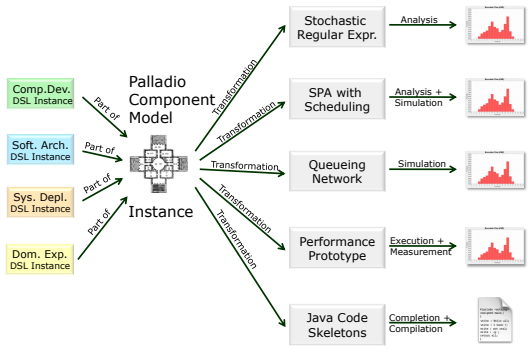
[Cheseman2000, Koziolek2006a]

Developer Roles



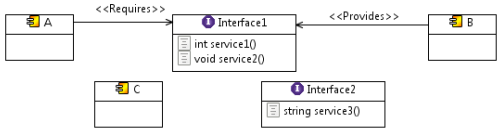


[Becker2007a]



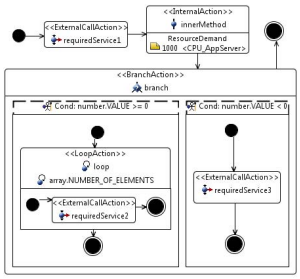
[Becker2007a]

Repository



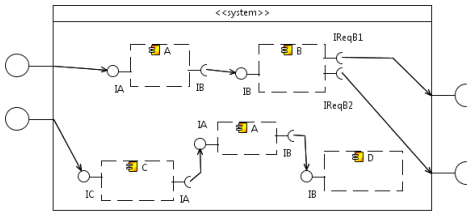
Component Developer

Service Effect Specification



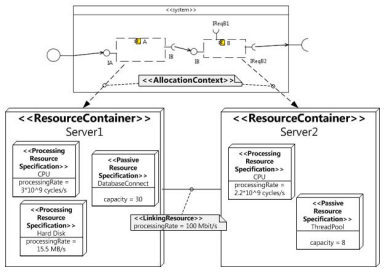
Component Developer

System



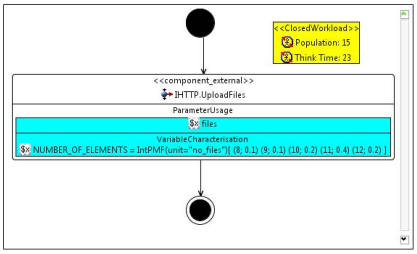
Software Architect

Resource Environment



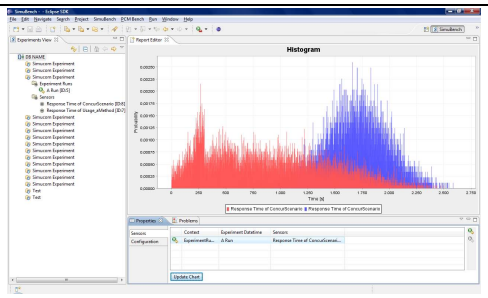
System Deployer

Usage Model



Domain Expert

Model Solver



Model Solver

PCMSolver

- Only Single User
- Fast (~2-5 seconds)
- Analytical Method, High Precision
- Stochastic Process Algebra based
- Traverses the architecture once
- Directly convolutes specified probability functions

SimuBench

- Single + Multiple User
- Slow (~30-600 sec.)
- Process-based Simulation
- Queueing Network based (G/G/n)
- Traverses the architecture repeatedly
- Draws samples from probability functions, adds them up

Example

- Blog-System ☺
- Switch to Eclipse!

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
3. Stochastical Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

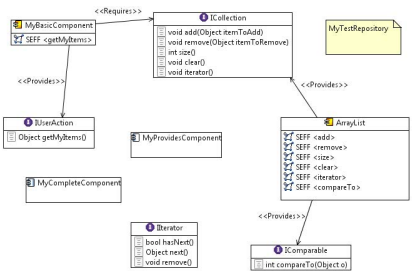
Lecture 1 / Lecture 2 / Lecture 3

Tasks

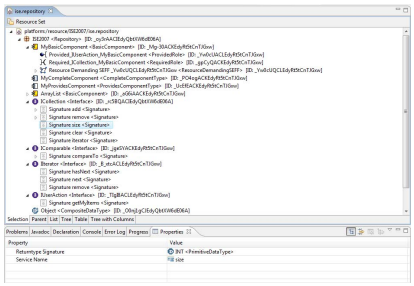
- Specifies Component & Interfaces
- Specifies Data Types
- Builds Composite Components
- Creates Service Effect Specifications
- Stores Modelling & Implementation Artefacts in Repositories
- Implements Components
- Tests Components
- Maintains Components

Component Developer

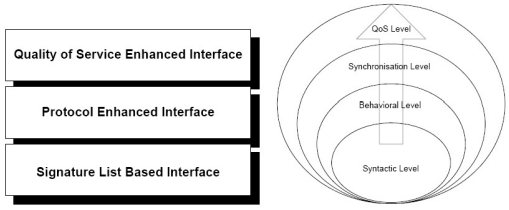
Example Repository



Example Repository



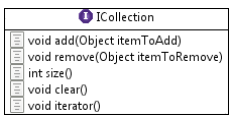
Interfaces



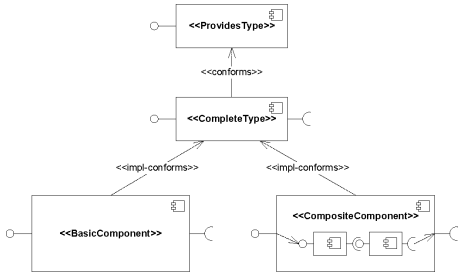
[Beugnard1999]

PCM Interfaces

- QoS (=Performance, Reliability)
  - Service Effect Specification (Lecture 2)
- Protocol (=Valid Call Sequences)
  - Finite State Machine (Not shown here)
- Signature
  - Corba IDL:
    - Return Type
    - Name
    - Parameter List
    - Exception List

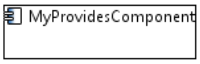


PCM Component Types



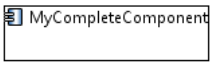
Provided Component Type

- Only Provided Interfaces mandatory
- May contain required services, not mandatory
- Specified during early development, refined later
- Situation: certain functionality needed, but additionally required services unknown
- QoS Annotations



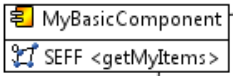
Complete Component Type

- Provided and Required Interfaces mandatory
- Dependencies between Provided and Required Interfaces not fixed
- Situation: Needed functionallity known, component environment also fixed



Basic Component

- Provided/Required Interfaces mandatory
- One option to implement a Complete Type
- Service Effect Specification for Dependencies between Provided and Required Interfaces
- May be composed to Composed Components



Composite Component

- During development composed from any component types
- Finally composed from Basic Components and/or other Composite Components
- Likely not used in the experiment, but may occur in exercises

Data Types

- Primitive Datatype
  - INT, CHAR, BOOL, DOUBLE, LONG, …
- Collection Datatype
  - Contains an inner primitive datatype
  - ARRAY, SET, LIST, TREE, HASHMAP, …
- Composite Datatype (Struct)
  - Contains inner primitive and/or collection and/or composite datatypes
  - ADDRESS, CUSTOMER, PERSON, …

Hands on Example

- Switch to PCMBench

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)

   Lecture 1
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF

   Lecture 2
3. StoEx
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

   Lecture 3

Lessons Learned Today

- Person – Role – Task
- Component Developer, Software Architect, System Deployer, Domain Expert
- PCMSolver vs. SimuBench
- PCM Repository (Component Developer)
  - Components (Provided, Complete, Basic, Composite)
  - Interfaces (Signature, Protocol, SEFF)
  - Data Types (Primitive, Collection, Composite)

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part II (PCM)

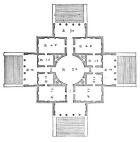
Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
   — Lecture 1
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF — Lecture 2
3. Stochastical Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies
   — Lecture 3

Service Effect Specification

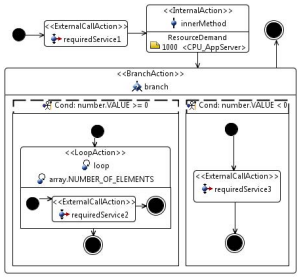
- Description of the external visible actions of a component's service
- Abstraction of internal behaviour
- Describes relationship between provided component side and required component side
- Can be parameterised by variables (see next lecture)



Conceptual Sources of the SEFF

- CBSE Parametric Contracts
- UML2 Activities
  - Notation
  - Some semantic ideas
- Software Execution Graphs of SPE
- Core Scenario Model (CSM) used in PUMA (Performance by unified model analysis)
- KLAPER (Kernel Language for Performance and Reliability Analyses)

Service Effect Specification Overview



Component Developer

Conceptual Overview

- Resource Actions
  - Internal Action
  - Acquire- & Release Action
- Communication
  - External Call Action
- Control Flow
  - Loops
  - Branches
  - Fork

Start and Stop Action

- Mark beginning and end of activities
- Every sub activity also has to have one start and one stop action

Internal Action

- Modells component internal activities like doing a computation
- Specifies the summed up resource demand for the action
- Different resources can be used

Acquire Action

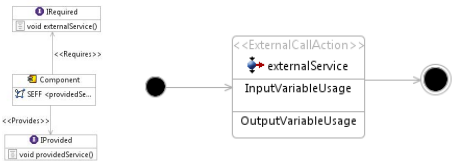
- Acquire Actions model the acquisition of a limited resource (Passive Resource Type)
- Examples are Database Connections, Pooled Threads, Mutex Locks, …
- Serve as synchronisation mechanism for concurrent executions

Release Action

- Release acquired resources again
- Other waiting jobs can use the resource now
- A FIFO strategy controls the order of acquisition for the waiting jobs

ExternalCallAction

- Models a call using any of the required roles
- A call *must* use a required role
- Parameter passing and returning can be specified (next lecture)



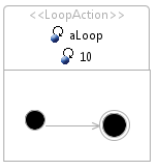
Control Flow Specification

- Control flow constructs model the course of actions like in SPE
- Concepts available
  - Loops
    - Loop
    - CollectionIterator
  - Branches
    - Probabilistic
    - Guarded
  - Forks

Loops

- Models repeated behaviour
- Iteration count has to be specified explicitly

```
for(int i=0; i<10; i++){
    …
}
```

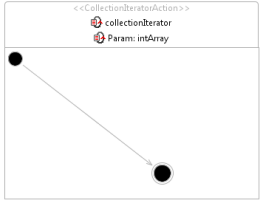


CollectionIteratorAction

- CollectionIteratorActions iterate over all elements in an instance of a CollectionDataType
- The behaviour is executed for every element

CollectionIteratorAction

```
void myMethod(int[] intArray)
{
   for (int x:intArray) {
      do
      …
   }
}
```



Semantic details

- Loop and CollectionIterator semantics preview
  - Inner Actions are evaluated *stochastically independent* wrt. to contained parametric dependencies
  - Collection Iterator Actions are evaluated *stochastically dependent* wrt. to the characterisation of the parameter being iterated
- Examples and further details in next lecture

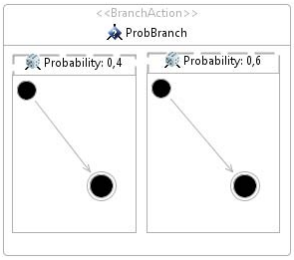
Branches

- A branch models optional parts of the control flow
- Exactly one branch must be executed, to model an option an empty alternative branch has to be specified
- Two flavours:
  - Probabilistic Branch Transitions: A probability can be specified for every branch which is the probability of executing the branch. Probabilities have to sum up to 1
  - Guarded Branch Transitions: Guards „protect" the execution of the branch. Execute branch which guard is true

Probabilistic Branches

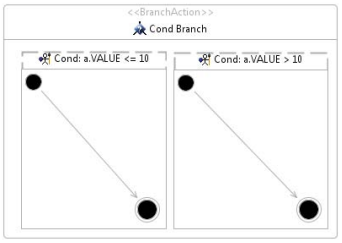
```
If (someCondition) {
      …
}
else{
      …
}

someCondition == true in
40% of all cases
```



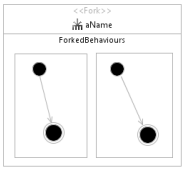
Guarded Branches

```
a = …
If (a <= 10) {
      …
} else {
      …
}
```



Fork

- A fork spawns n threads and waits for them to finish
- After finishing the forked threads the main thread continues



Now: Exercises in the Tool

- Switch to Eclipse!

Lessons Learned Today

- What is a SEFF?
- What is it used for?
- Concepts
  - Resource Actions
    - Internal Action
    - Acquire- & Release Action
  - Communication
    - External Call Action
  - Control Flow
    - Loops
    - Branches
    - Forks

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part III (PCM)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
   — Lecture 1
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
   — Lecture 2
3. Stochastic Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies
   — Lecture 3

Uncertainties

- A situation is uncertain if the outcome is unknown in advance
- Probabilistic characterisations possible
- Examples
  - How will users interact with a system?
  - When do they arrive?
  - Which parameters do they pass in their calls?

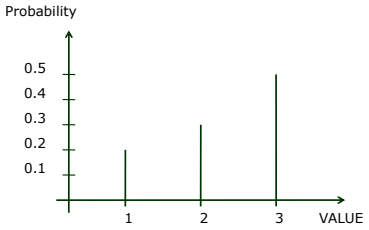
Random Variables

- Random variables describe uncertain events
- They may be described by their probability distribution
- Two kinds of random variables:
  - Discrete
    - Example: Iteration count of a loop
  - Continuous
    - Example: Passed time between the arrival of two jobs

Probability Mass Function

- PMF
- Distribution Function of a discrete variable
- Domain type depends on the model
  - Loop Iterations: Integer
  - Collection Structure: Enum
  - Actual Value: Any
  - …
- PMF Literals
  - IntPMF[(1;0.1)(2;0.3)(5;0.6)]
  - EnumPMF[(„Sorted";0.5)(„Unsorted";0.5)]
- Constraint: Sum of probabilities has to be 1, be careful, this is still unchecked in the tools!

Probability Mass Function

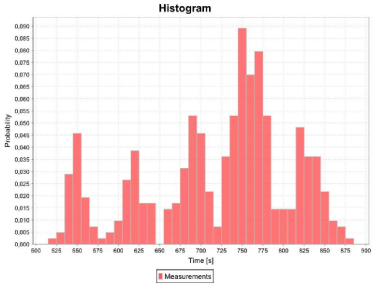


IntPMF[(1;0.2)(2;0.3)(3;0.5)]

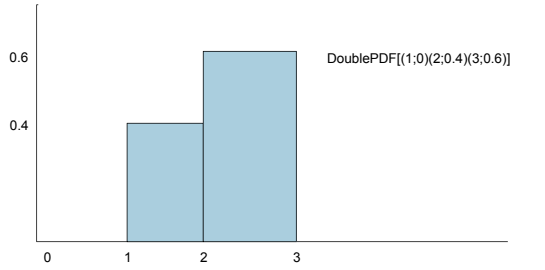
Probability Density Function

- PDF
- Dist. Function of a continuous random variable
- Domain is always double
- Hard to characterise as possibly infinite
  → We use a derived discrete function: BoxedPDF
- Boxes sum up all events falling into their bounds
- Inner box distribution is uniform
- Depicted as histogram or CDF

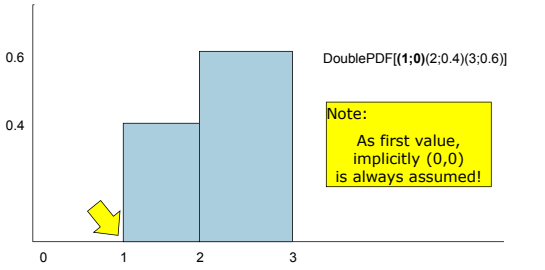
PDF



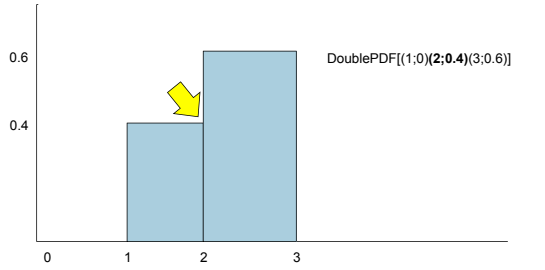
Specification



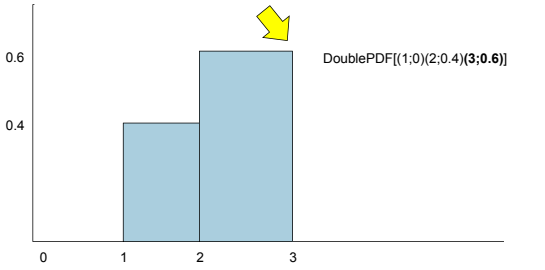
Specification



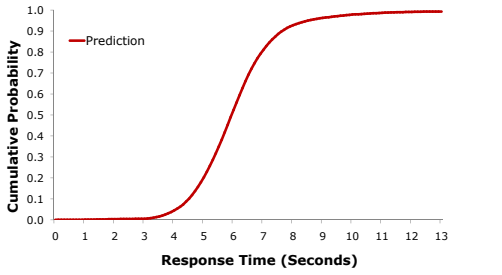
Specification



Specification



PDF



Semantics

- X ~ DoublePDF[(1;0)(2;0.4)(3;0.6)]
  - P(0<=x<1) = 0
  - P(1<=x<2) = 0.4
  - P(2<=x<3) = 0.6
  - P(1<=x<1.5) = 0.2
  - …
- Y ~ IntPMF[(1;0.2)(2;0.5)(3;0.3)]
  - P(Y = 1) = 0.2
  - P(Y = 2) = 0.5
  - P(Y = 3) = 0.3
  - P(Y = n) = 0 for all n >=4

Functional dependent random variables

- X ~ IntPMF
- Y ~ IntPMF
- Z = X * Y
  - Z is also a Random Variable
  - Z ~ IntPMF
  - Z's distribution is derived automatically
- Operators: +, -, *, /, ^, <, >, …
- Resulting grammar is called Stochastic Expression (StoEx) in PCM

Using random variables for modelling

- Where can we use random variables?
  - Loop iterations
  - Branch conditions
  - Inter arrival time
  - Think time
  - For input parameter characterisations
  - For output/return parameter characterisations
  - For resource demands

Introducing variables...

- We can define our own variables to describe parameters
- They are set at the caller's side
- They are used at the called side
- Model performance relevant dependencies *only!*
  - Most parameters have no or only little influence on the performance
  - Omit these parameters from the specification!
  - Example: int ICalculator.add(int a, int b) Performance is not depending significantly on any parameter value!

Parameter Abstractions

- We normally do not model parameter values but performance abstractions
- The following types are available
  - BYTESIZE: Memory footprint of a parameter
  - VALUE: The actual value of a parameter for primitive types
  - STRUCTURE: Structure of data, like „sorted" or „unsorted"
  - NUMBER_OF_ELEMENTS: The number of elements in a collection
  - TYPE: The actual type of a parameter (vs. the declared type)

Examples

Void aMethod(int a, int[] b, MyFigure c)

Caller Specifies:

a.BYTESIZE = 4

a.VALUE = IntPMF[(10;0.2)(30;0.4)(100;0.4)]

b.NUMBER_OF_ELEMENTS = 100

c.TYPE = EnumPMF[(„circle";0.4)(„rectangle";0.6)]

Example cont.

Void aMethod(int a, int[] b, MyFigure c)

Use in the SEFF of aMethod

aLoop.Iterations = a.VALUE

anAction.ResourceDemand = b.NUMBER_OF_ELEMENTS * 100

aBranch.Condition = c.TYPE == „circle"

Special Keywords

- INNER
  - Refers to the elements of a collection
  - Describes the contents of the collection
- RETURN
  - Refers to the return value of the current SEFF
  - Characterises the result
- Namespace of variables
  - Characterise inner elements of composed data types

Examples

Void aMethod(int a, int[] b, MyFigure c)

b.INNER.BYTESIZE = 4

b.INNER.VALUE = 42

b.INNER.VALUE = IntPMF[(42;0.5)(43;0.5)]

c.color.VALUE = EnumPMF[(„red";0.1)(„green";0.9)]

Editor Support

- „StoEx-Dialog"
- Offers syntax highlighting, code completion, online help and basic syntax checking
- Often available on double click of the corresponding model element

Semantic difference Loop and Collection Iterator

- In a Loop *all* characterisations are evaluated any time they occur (stochastical independence)

```
// a.INNER.BYTESIZE=IntPMF[(1;0.5)(10;0.5)]
Object[] a = ...
for (int i=0; i < 10; i++) {
  // a.INNER.BYTESIZE can be 1 in doSth
  doSth(a[i]);
  // a.INNER.BYTESIZE can be 10 in doSthElse
  doSthElse(a[i]);
}
```

Semantic difference Loop and Collection Iterator

- In a Collection Iterator *all* characterisations are evaluated any time they occur (stochastical independence) execpt the INNER characterisations of the iterator parameter

```
// a.INNER.BYTESIZE=IntPMF[(1;0.5)(10;0.5)]
Object[] a = ...
for (Object o:a) {
  // a.INNER.BYTESIZE can be 1 in doSth
  doSth(o);
  // a.INNER.BYTESIZE is also 1 in doSthElse
  doSthElse(o);
}
```


Semantics: Dependant Branches

```
// x.VALUE=IntPMF[(1;0.5)(6;0.3)(12;0.2)]
if (x > 5) {
  if (x > 10) {
  } else {
  }
}
```

If you would have to model this with probabilistic branch transitions, what would be the probabilities? (Tip: Bayes Theorem!!!)

Semantics: Dependant Branches

```
// x.VALUE=IntPMF[(1;0.5)(6;0.3)(12;0.2)]
if (x > 5) { // p = 0.5
  // x.VALUE is always 6 or 12 here!
  if (x > 10) { // p = 0.4
     // x.VALUE is always 12 here!
  } else { // p = 0.6
     // x.VALUE is always 6 here!
  }
} else { // p = 0.5
  // x.VALUE is always 1 here!
}
```

Our tools respect this automatically, you don't have to calculate on your own!

Now: Exercises in the Tool

- Switch to Eclipse!

Lessons Learned Today

- What is uncertainty?
- How is it modelled in PCM?
- Random Variables
- Random Variables in the PCM
  - Loop Iterations
  - Branch Conditions
  - Resource Demands
  - Parameter characterisations
  - Usage model details

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part IV (PCM)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
3. Stochastic Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

Lecture 1 / Lecture 2 / Lecture 3

Outline

4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation
6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Solver, Result Interpretation
8. Comprehensive Case Study
9. Outlook

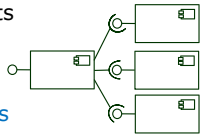
Lecture 4 / Lecture 5

Software Architect

[http://commons.wikimedia.org/wiki/Image:Architect.png]

Software Architect: Tasks (1/2)

- Specifies an architecture (boxes and lines) from existing components and interfaces
- Specifies new components and interfaces
- Uses architectural styles and architectural patterns
- Analyses architectural specification and makes design decisions

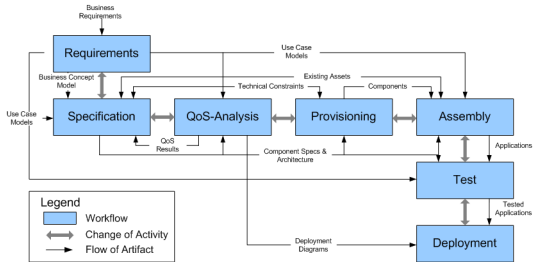


Software Architect: Tasks (2/2)

- Conducts performance prediction based on architectural specification
- Delegates implementation tasks to component developers
- Guides the whole development process

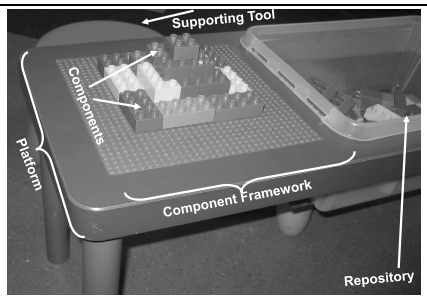


CBSE Development Process



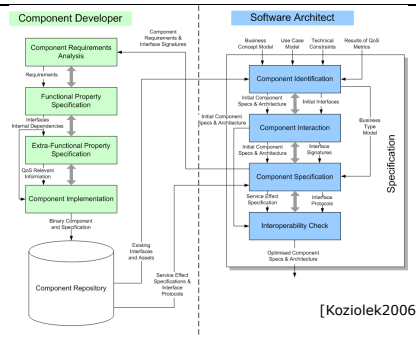
[Cheeseman2000, Koziolek2006a]

Specification Process



[Grunske2007]

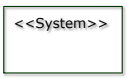
Specification Process



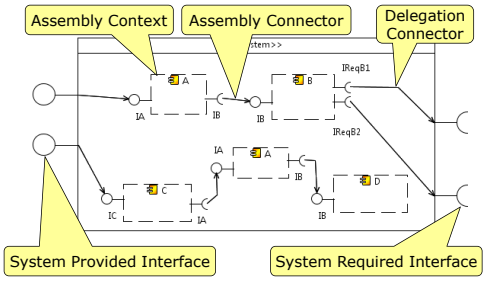
[Koziolek2006a]

System

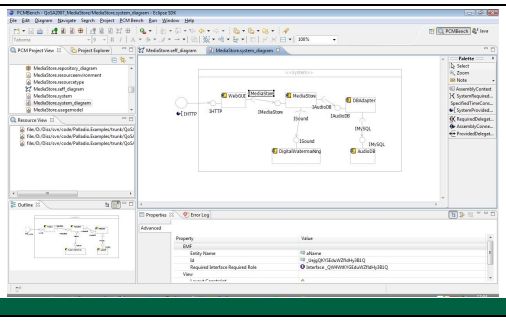
- Models the component-based architecture to be analysed
- May include components from different repositories
- Provides an interface for users
- Excludes uninteresting services and connects to them via system required interfaces
- Is a prerequisite for the system deployer to allocate the components



System Specification

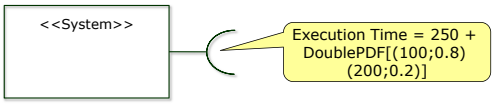


System Specification PCM Bench

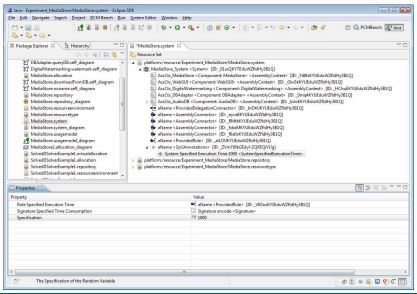


QoS Annotation

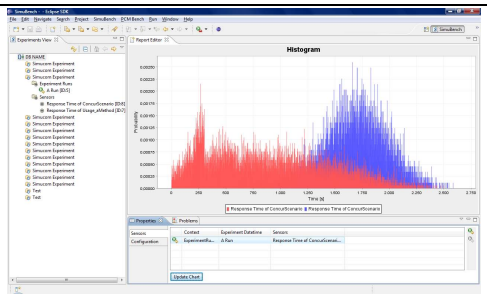
- System Required Interfaces: connection to functionality not modelled in the system
- Example: web service, unknown component
- Execution time specification necessary



QoS Annotation



Performance Evaluation



Design alternatives changing performance

- More hardware
- Faster hardware
- Caching
- Resource Pooling
- Replication
- Load Balancing
- Compression
- Reducing communication overhead
- Reimpl. of a component
- Allocation
- Introduce parallel processing
- Use Performance Pattern
- ...

Outline

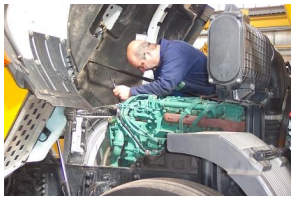
4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation

Lecture 4

6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Solver, Result Interpretation
8. Comprehensive Case Study
9. Outlook

Lecture 5

System Deployer



[http://www.dorsetforyou.com/]

System Deployer: Tasks

- Models the resource environment (e.g., middleware, OS, hardware)
- Models the allocation of components to resources
- Sets up the resource environment (e.g., installing application servers, configuring hardware)
- Deploys components on resources (e.g., writing deployment descriptors)
- Maintains the running system

Resource Types

- Abstract specification of resources (e.g. CPU, HD, Net)
- Why?
  - concrete resources (e.g. 2 GHz CPU, 20 MB/s HD, 1 Gbit/s Net) unknown during component specification and implementation
- Thus: component developers provide RDSEFF specifications referring to resource types
- Once the concrete resource environment is specified, timing values can be derived

Resource Types in PCM

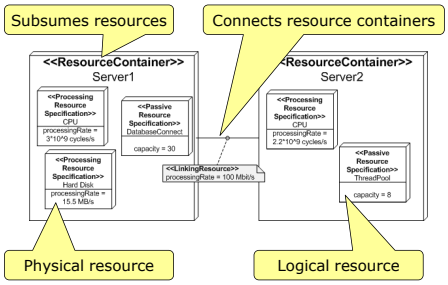
CPU

HD

Network

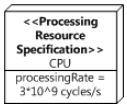
Memory

Resource Environment



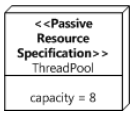
Processing Resources

- Model CPUs, Hard Disks, Networks, etc.
- Specify a processing rate for the resource demands of the RDSEFFs
- Example 1:
  - Processing rate (CPU): 3*10^9 cycles/s = 3 Ghz
  - RDSEFF: Resource Demand = 1,5 * 10^9 cycles
  - ➔0,5 seconds execution time
- Example 2:
  - Processing rate (HD): 20 MB/s
  - RDSEFF: Resource Demand = 500 000 Byte
  - ➔0,025 seconds execution time

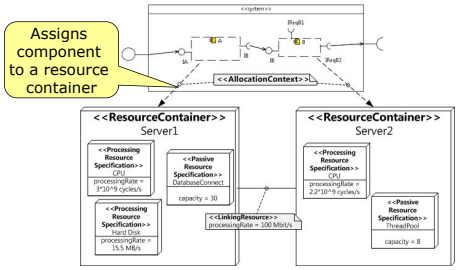


Passive Resources

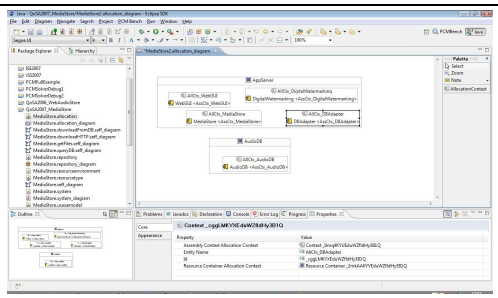
- Model logical resources
  - Threads, Semaphores, Database connections, …
- Are aquired or released in RDSEFFs
- Specify a maximum capacity
- Example:
  - Capacity (ThreadPool): 8
  - RDSEFF: AquireAction(ThreadPool)
  - ➔Afterwards: #available threads decreased by 1
  - RDSEFF: ReleaseAction(ThreadPool)
  - ➔Afterwards: #available threads increased by 1



Allocation



Allocation



Outline

4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation

Lecture 4

6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Solver, Result Interpretation
8. Comprehensive Case Study
9. Outlook

Lecture 5

Lessons Learned Today

- Software Architect
  - Specification of a system
- System Deployer
  - Resource Types
  - Specification of a resource environment
  - Specification of an allocation

Switch to Eclipse

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part V (PCM)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation

Lecture 4

6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Result Interpretation

Lecture 5

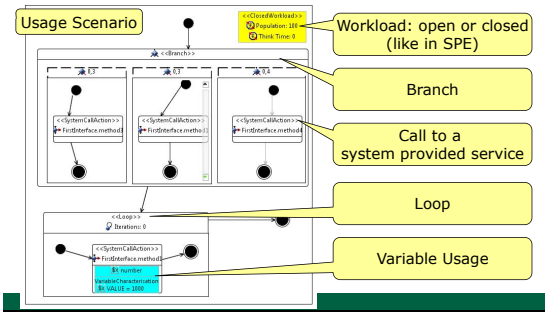
Domain Expert

- Familiar with the business domain
- Specifies user behaviour
  - Number of users
  - User Requests to the System
  - Input parameters characterisations as distribution functions

Usage Model

- Models ***user*** behaviour, not component!
- Similar to RDSEFFs, but
  - Does not refer to resources
  - Does not refer to inner components of a system
  - Does not model parametric dependencies
  - Includes a workload specification
- Usage Model
  - *1…n* usage scenarios (1 per use case)
  - *1* workload per usage scenario

Usage Model



Outline

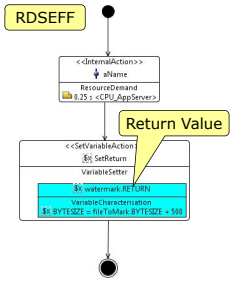
4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation

Lecture 4

6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Result Interpretation

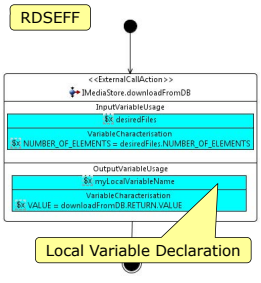
Lecture 5

SetVariableAction



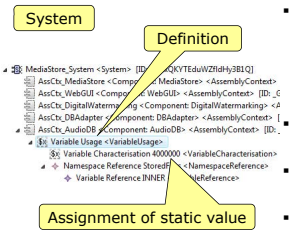
- Characterisation of Return Values
- Only if performance relevant!
- Reserved Keyword RETURN
- May occur in different branches

Using Return Values of ExternalCallActions



- Assignment of output parameter characterisations to local variables
- Use local variables afterwards in parametric dependency specification

Component Parameters



- Global parameters for components
  - Configuration options
  - Static State
  - ...
- Declaration per assembly context
- Default value by component developer
- Cannot be changed dynamically (during simulation)

Model Validation

- Switch to Eclipse!

Outline

4. Software Architect
   a) System (Composed Structure)
   b) QoS Annotations on System Interfaces
5. System Deployer
   a) Resource Types, Resource Environment
   b) Allocation

Lecture 4

6. Domain Expert
   a. Usage Model
   b. Parameter Characterisations
7. Result Interpretation

Lecture 5

Result Interpretation

- Performance Metrics PCM
- Statistics
- Analysing Histograms
- Analysing Cumulative Distribution Functions

Performance Metrics supported by SimuCom

✔ Response Time per Time Sensor
- Histogram
- Cumulative Distribution Function
- Point Estimators with R (Statistics Package)

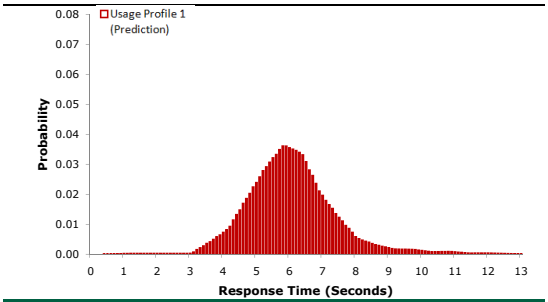
✔ Utilization per Resource
- Percentage: Busy Period / Idle Period

✘ Currenty NOT supported
- Throughput

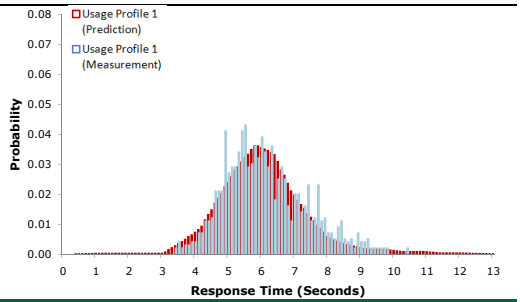
Statistics

- Point Estimators
  - Expected Value (Mean)
  - Standard Deviation
  - Variance
  - Median
- Compare Probability Distributions
  - Kolmogorov-Smirnov-Test
  - Chi-Square-Test
  - Anderson-Darling-Test

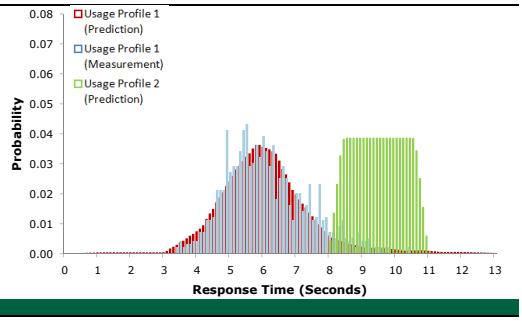
Result Interpretation



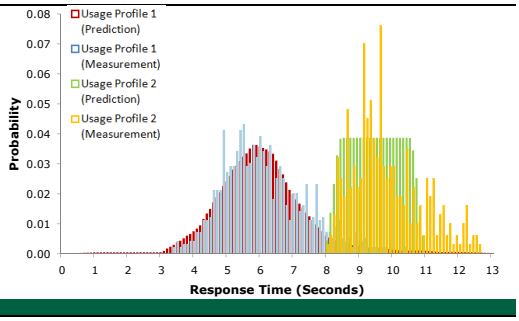
Result Interpretation



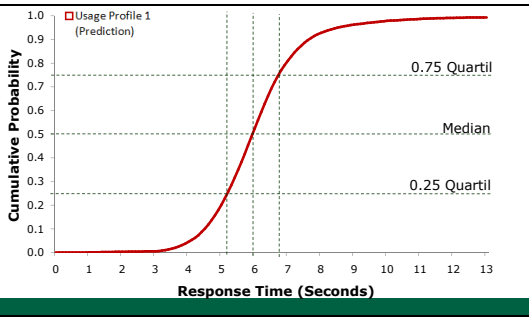
Result Interpretation



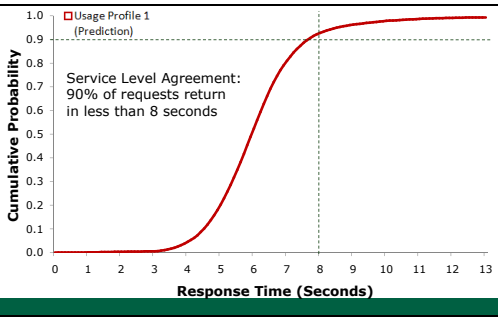
Result Interpretation



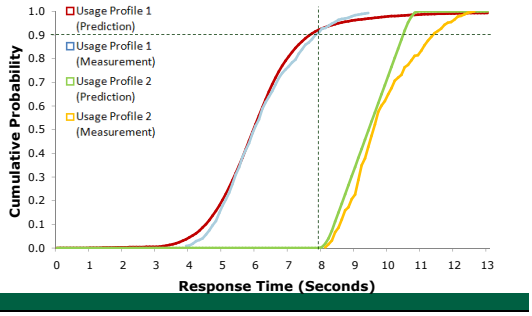
Result Interpretation



Result Interpretation



Result Interpretation



Current Developments (Changelog)

- Linking Resources
  - work automatically in background
  - latency specification for comm.link.resources
- Scheduling Policies for ProcessingResources
  - FCFS, PROCESSOR_SHARING, DELAY
- System
  - Output parameters for system external calls
  - Broker lookup support for connectors
- Usage Model
  - User Delays (to model waiting/thinking)

Current Developments (Changelog)

- Stochastical Expressions
  - AND, OR, NOT for Boolean Expressions
  - Standard Probability Distributions
    - Exp(x), UniForm(x,y), Norm(x), …
- OCL constraints for model validation
- SimuCom
  - Saving simulation results to disk

Lessons Learned Today

- Usage Model
  - for user behaviour
- Return Values
- Component Parameters
- Model Validation
- Result Interpretation
  - Probability distributions
  - Point estimators

A.4 Review Slides

The last tutorial session before the experiment reviewed aspects of both methods and was created by the author. The last slides were presented after the experiment, presenting first results.

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Übung: Experimentvorbereitung

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

ZUR ÜBUNG

Passive Ressourcen

- In SPE-ED kein Acquire-Release
- Wartezeit abschätzen.

- Für eine passive Ressource:
  - Wahrscheinlichkeit zu warten ist Auslastung der Ressource
  - p(belegt) = Verweilzeit * Ankunftsrate
  - p(belegt) = 0.01 * 0.3 = 0.003
- Wie lange warten?
  - Wartezeit ≈ Verweilzeit (geringe Auslastung)

Abschätzung passiver Ressourcen

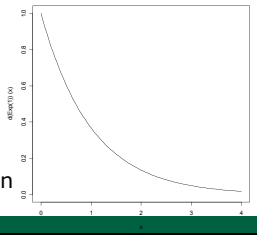
- Für mehrere Ressourcen, z.B. Threadpool 8
  - Anzahl Ressourcen n
  - Erst warten wenn letzte Ressource auch belegt
  - wenn n Benutzer gleichzeitig im System
  - wenn Ankunftsrate * Verweilzeit > n
  - wenn Verweilzeit > n / Ankunftsrate
  - und das für n Benutzer hintereinander

Abschätzung passiver Ressourcen

- Verweilzeit > n / Ankunftsrate für n Benutzer hintereinander
  - Abschätzen: p(Verweilzeit > n / Ankunftsrate)
  - Für n Benutzer hintereinander:
    - p(Verweilzeit > n / Ankunftsrate)^n
- Beispiel: Threadpool 8, Ankunftsrate 1 User/s
  - Durchschnittliche Verweilzeit: 3 s
  - Abschätzen: p(Verweilzeit > 8) ≈ 0.05
  - Dann (p(Verweilzeit > 8))^8 = 3,9 * 10^{-9}

Ungenauigkeit

- Nicht betrachtet: Ankunftsrate ist nicht fix
  - Es kommt nicht genau jede Sekunde ein Benutzer am System an
- Ankunftsrate ist exponential-verteilt
- Mit Erwartungs-wert 1
- Auch Möglichkeit, dass 8 Benutzer in 3 s ankommen.

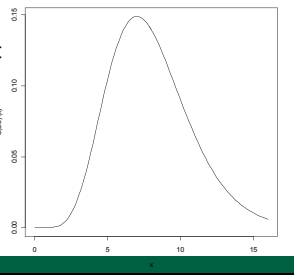


Ungenauigkeit

- Möglichkeit, dass 8 Benutzer in 3 s ankommen
- Berechnung der Wahrscheinlichkeit durch Faltung der Funktionen

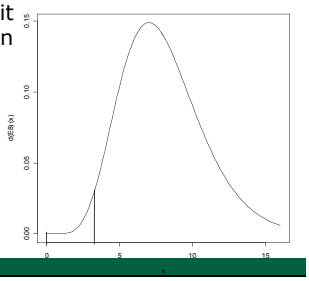
$$(f * g)(t) = \int_D f(\tau) g(t - \tau)\, \mathrm{d}\tau$$

- Dichtefunktion für die Ankunftszeit von 8 Benutzern



Ungenauigkeit

- Man findet heraus:
- Wahrscheinlichkeit dass 8 Benutzer in 3 s ankommen: 0,012



Passive Ressourcen in SPE-ED

- Einflüsse:
  - Benutzer können länger bleiben als im Mittel
  - Benutzer können schneller ankommen als im Mittel
- Beide Einflüsse hier einzeln untersucht.
  - Müssten eigentlich zusammen betrachtet werden!
- Einfluss Ankunftsrate deutlich höher.
  - Aber nicht von Hand berechenbar
- Diese Daten als Grundlage für Schätzungen verwenden.
  - Grundlagen für Schätzungen angeben.

Ankunftsraten

- In SPE-ED als Ankunftsrate angegeben
  - Nicht fix, sondern immer exponentialverteilt
- In Palladio als Zwischenankunftszeit
  - StoEx
  - Fixe Werte im StoEx sind feste Ankunftszeiten
  - Auch Verteilungen im StoEx möglich
- Gleiche Voraussetzungen beider Verfahren
  - In Palladio Exp(Ankunftsrate) im StoEx

weiter

Angaben zu Geräten

- SPE-ED Angabe einer Service Time
  - Wie lange benötigt Device, um eine Service Unit abzuarbeiten
  - Z.B. Service Unit = Work Unit, Service Time 0.001 **s/WU**
- In Palladio Angabe einer Processing Rate
  - Wie viele Service Units bearbeitet Device pro Sekunde
  - Z.B. Service Unit = Work Unit, Processing Rate 1000 **WU/s**

Service Units in Palladio

- In den Übungen wurde teilweise als Einheit für Resource Demands „ms on“ angegeben
- Verletzt Unabhängigkeit zwischen den Rollen!
- Abstraktere Einheiten angeben
- Nur eine Service Unit pro Gerät
- Dadurch Entscheidung ob Latenz oder Durchsatz.

Service Units in Palladio

- Für eine Festplatte kann Latenz (Seek Time) oder Durchsatz spezifiziert werden.
- Durchsatz:
  - Resource Demands in Service Unit Byte angeben
  - Processing Rate in Byte/s
- Latenz / Seek Time
  - Resource Demands in Number of Accesses
  - Processing Rate in NoA/s

→ Beispiel im PCM

Service Units in Palladio

- Für Linking Resources kann Durchsatz und Latenz angegeben werden.
- Erinnerung: Die LinkingResource wird automatisch belastet
  - Aufruf einer Komponente über Resource Container Grenzen und
  - Wenn Aufruf oder Rückgabewert eine .BYTESIZE Charakterisierung haben

Vorgehen Palladio

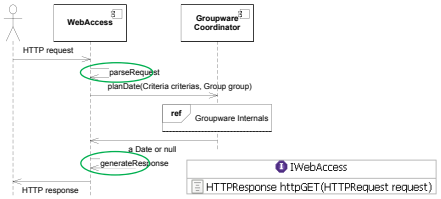
- An den verschiedenen Rollen des PCM orientiert
- Komponentenentwickler
- System Architect
- Deployer
- Domain Expert

Komponentenentwickler

- Repository und Schnittstellen
- SEFFs
- Parametrisierungen
- Eine Vorgehensweise für ein gesamtes Repository
  - Zunächst Kontrollfluss modellieren
  - Interne Aktionen die Parametrisierung benötigen identifizieren
  - Parameter durchleiten oder als Komponentenparameter erwarten

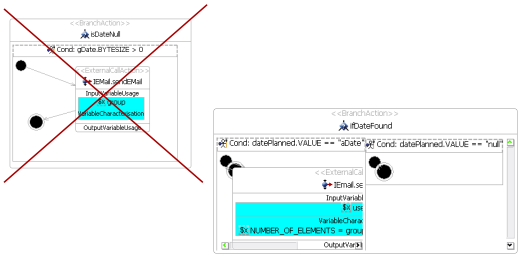
Repository

- Interne Methode werden nicht in die Schnittstellen aufgenommen
- Nur Hinweis auf interne Aktionen



PCM Branch Actions

- Vollständige Abdeckung aller Alternativen.



Rückgabewerte

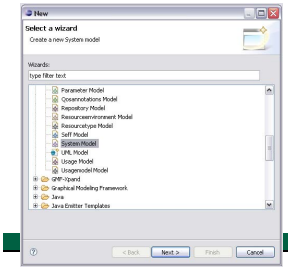
- Werden mit SetVariableAction gesetzt
- Haben anderen Kontext (Stackframe) als lokale Parameter
- Sprich: Gesetzte Return Werte können im SEFF selbst nicht verwendet werden!
  - Nur das Überschreiben ist möglich.

Software Architekt

- Spezifiziert, wie Komponenten zum System zusammengesetzt werden
- AssemblyContext
- Setzt Komponentenparameter für einzelne Assemblies

System in Palladio

- System Model manuell initialisieren
- Composite Diagramm darauf initialisieren



Vorteile:
- Assembly Contexts werden benannt
- Weniger Bugs

System Diagram wird nicht länger gepflegt

System in Palladio

- System Models können kopiert und für Entwurfsalternativen angepasst werden
- Nach dem Kopieren neues Composite Diagram auf der Kopie erzeugen
- Man muss nicht immer alles neu verbinden

Deployer

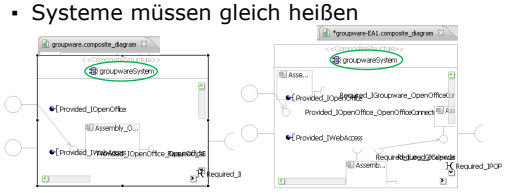
- Spezifiziert die Ressourcenumgebung
  - Processing Rate
- Spezifiziert, welche Komponenten wo eingesetzt werden
- Genauer: Welcher Assembly Context wo eingesetzt wird
- Bildet Assembly Contexte auf Server (=Resource Container) ab

Domänenexperte

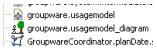
- Modelliert das Benutzungsprofil.
- Liefert Werte für die in den SEFFs benötigten Parametrisierungen
- Bestimmt die Ausführungshäufigkeit
  - Closed Workload: X Benutzer zirkulieren im System
  - Open Workload: Zeit die zwischer den Ankunft zweier Benutzer vergeht (nicht Ankunftsrate!)
- Bestimmt das Verhalten der Benutzer
  - Beispielsweise rufen die Benutzer immer erst Methode A und dann zu 30% auch Methode B auf

Usage Model für mehrere EAs

- Systeme müssen gleich heißen



- Dann kann ein Usage Model mit beiden verwendet werden



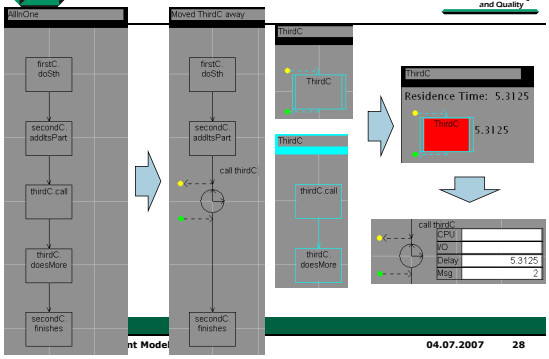
Wie in SPE-ED umsetzen

- Keine Rollentrennung
- Kontrollfluss modellieren
  - Komponentenentwickler, Systemarchitekt
- Resource Demands angeben
  - Komponentenentwickler, Domänenexperte
- Umgebung angeben als Overhead Matrix
  - Deployer

Deployment in SPE-ED

- Ein Szenario auf eine Facility deployed
- Anspruch: Gesamter Kontrollfluss
- Verteilte Systeme:
  - Teile des Szenarios auslagern, die auf anderer Facility ablaufen
  - Weitere Facility dafür auswählen
  - Antwortzeit für ausgelagerte Szenarien bestimmen
  - Synchronisationsknoten mit entsprechendem Delay versehen

Beispiel



UMGANG MIT DEN TOOLS

SPE-ED

- Begrenzte Anzahl Navigationsboxen
  - Vorher überlegen, wie viele Boxen benötigt werden
  - Inhalte von Schleifen ggf. zusammenfassen.
- Simulation sehr unbeständig
  - Wenn möglich Contention Solution verwenden
  - Mehrere Use Cases in ein Szenario
  - Mit Branches auswählen
  - Quasi Benutzungsmodell mit ins Szenario

Palladio

- Nur ein Diagramm offen haben!
- Composite Diagram, nicht System Diagram
- Möglichkeit, Ergebnisse zu persistieren (DB4O)
- Wenn ein Diagramm defekt ist, nur Diagramm aus Modell neu erzeugen
- System Modelle können kopiert werden, Diagramme nicht

WEITERE HINWEISE

Komplexe Benutzungsprofile

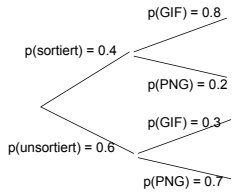
- Benutzungsprofile bisher mit unabhängigen Charakterisierungen
- Sind Einflussfaktoren auf die Performanz
- Verschiedene Charakterisierungen können aber auch voneinander abhängen
- Beispiel:
  - Liste von Bildern soll sortiert und zu einem Bild zusammengefügt werden
  - Zwei Charakterisierungen: Sortierung und Grafikformat

Beispiel Benutzungsprofil

- Liste von Bildern soll sortiert und zu einem Bild zusammengefügt werden
- Aufwand zum Zusammenfügen unterschiedlich je nach Grafikformat und ob sortiert oder nicht
- BP: 40 % sortiert, davon 80% GIF
  60 % unsortiert, davon 30 % GIF
- Verschiedene Handler für GIF und PNG

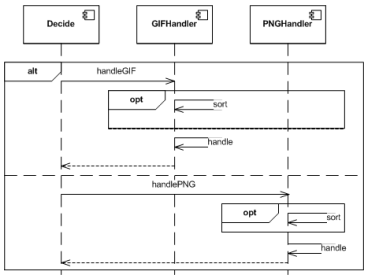
Wahrscheinlichkeiten

- Resultierender Baum



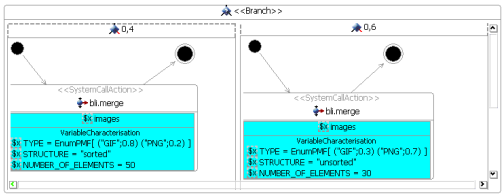
Kontrollfluss

- Ablauf



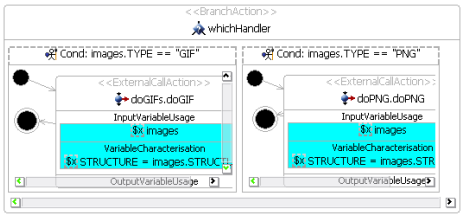
Palladio

- Im Benutzungsprofil Wahrscheinlichkeiten angeben
- Später benutzen



Palladio

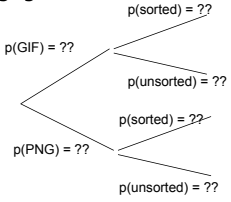
- In Abhängigkeit der Parameter modellieren



- Umkehr der Parameter stört nicht

SPE-ED

- Kontrollfluss entspricht nicht den angegebenen Wahrscheinlichkeiten



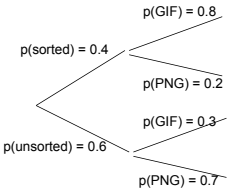
Wie ist Wahrscheinlichkeit für ein GIF?

Wie die Wahrscheinlichkeit dafür, dass sortiert werden muss, wenn bekannt ist, dass ein GIF vorliegt?

- Verschiedene Möglichkeiten zur Lösung

Wahrscheinlichkeiten

- Berechnung des ersten Schritts nicht schwer



p(GIF) = 0.4 * 0.8 + 0.6 * 0.3 = 0.5

p(PNG) = 0.4 * 0.2 + 0.6 * 0.7 = 0.5

Aber wie ist p(sorted | GIF) ?

Bayes Rule

- Zur Berechnung abhängiger Wahrscheinlichkeiten

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Vgl. auch Bayes-Theorem bei Wikipedia

- Hier also:

$$p(sorted|GIF) = \frac{p(GIF|sorted) * p(sorted)}{p(GIF)}$$

Bayes Rule

$$p(sorted|GIF) = \frac{p(GIF|sorted) * p(sorted)}{p(GIF)}$$

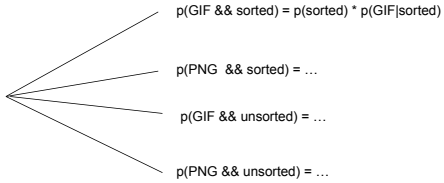
- Man erhält

$$p(sorted|GIF) = \frac{0.8 * 0.4}{0.5} = 0.64$$

- So für alle Äste der zweiten Ebene Wahrscheinlichkeiten bestimmen

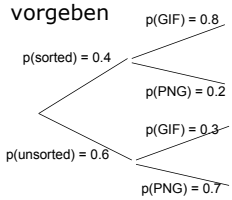
So kompliziert?

- Andere Optionen: Kontrollfluss nicht exakt nachbilden.
- Branch mit vier Optionen:



So kompliziert?

- Andere Optionen: Kontrollfluss nicht exakt nachbilden.
- Branches wie Wahrscheinlichkeiten sie vorgeben



Probleme

- Erst einmal leicht zu modellieren
- Wartbarkeit des Modells aber schlecht.
  - Insbesondere bei Komponenten
- Was wenn:
  - Eine Komponente vor den GIF Handler zwischengeschaltet wird?
  - Eine Komponente auf einen anderen Server deployed wird?
- Vor der Modellierung prüfen, ob solche Gefahren bestehen.

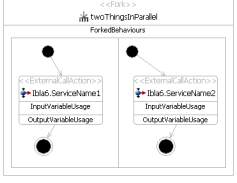
Parallelität in SPE-ED

- Kann nicht direkt modelliert werden
- Allgemein gilt für die Antwortzeit zweier parallel ausgeführter Aktionen A und B:
- Antwortzeit(A || B) = max(Antwortzeit(A), Antwortzeit(B))
- wenn unterschiedliche Ressourcen belastet werden.
- Für Aktion mit kleinerer Antwortzeit Resource Demand entfernen

→ Utilization ist nicht mehr aussagekräftig

Parallelität in Palladio

- Fork Action



- Ähnlich aufgebaut wie Branch
- Beide inneren Behaviours gleichzeitig ausführen

Nützliche StoEx Ausdrücke

- NOT
  - Beispielsweise für Branches
  - Branch 1: Date.VALUE == „null"
  - Branch 2: NOT (Date.VALUE == „null")
- Ternärer ?: Operator
  - Beispielsweise für Rechenaufwand
  - Group.NUMBER_OF_ELEMENTS > 10
    ? Group.NUMBER_OF_ELEMENTS
    : 10

Typsystem

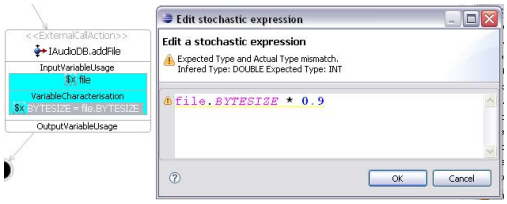
- Bytesize muss immer ein Int sein
- Integer können nicht als PDF modelliert werden
  - Aber wünschenswert um viele mögliche Ausprägungen zu beschreiben
- Funktion `Trunc()` um aus einer DoublePDF Integer zu ziehen.
- Auch um Bytesize zu verrechnen mit beispielsweise 0,5

Trunc() Beispiele

- BYTESIZE = Trunc(DoublePDF[…])
- BYTESIZE = Trunc(0.5 * sth.BYTESIZE)
- NUMBER_OF_ELEMENTS = Trunc(0.5 * sth.NUMBER_OF_ELEMENTS

Validierung Palladio

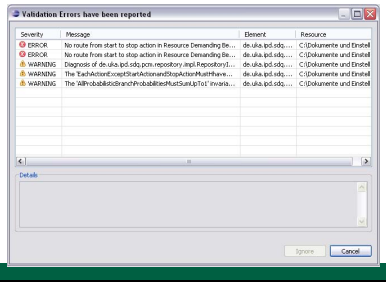
- StoEx nun auch Typechecking



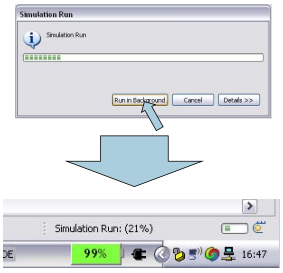
- Weitere Erweiterungen der Validierung

Validierung Palladio

- Direkt nach dem Start der Simulation



Simulation im Hintergrund



Weiterarbeiten ist möglich, es muss nicht gewartet werden

EXPERIMENTSITZUNG

Experimentsitzung

- Samstag, 9:00 - 14:00, Pool 356
- Notebook mitbringen (wer zugesagt hat)

\+ Butterbrezeln & Getränke, im Anschluss Pizza

Experimentsitzung

- Einteilung in zwei Gruppen
  - SPE-ED und Palladio
- Eine Aufgabenstellung
- Modellieren und Analysieren mit dem Tool
- Bei SPE-ED Facilities gegeben (wie gehabt)
- Bei Palladio Repository und Resource Environment gegeben

Universität Karlsruhe (TH)
Forschungsuniversität gegründet 1825

Praktikum Ingenieurmäßige Software-Entwicklung

Fragebögen, Erste Ergebnisse, Rückblick

Anne Martens / Ralf Reussner

Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Gliederung

- Erste Ergebnisse
  - Abschätzungen
  - Vorhersagen
  - „Musterlösung"
- Rückblick
- Fragebögen
  - Nachbereitung der letzten Experimentsitzung
  - Evaluation der Lehrveranstaltung

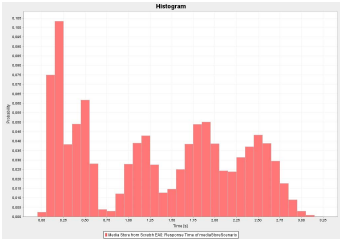
Erste Ergebnisse

Abschätzung Mediastore

Benutzungsprofil 1 (1 Benutzer)	Benutzungsprofil 2 (mehrere Benutzer)
1. Caching (EA 1)	1. Bitrate senken (EA 4)
2. Bitrate senken (EA 4)	2. Caching (EA 1)
3. DBC Pool (EA2) und 2. Server (EA3)	3. DBC Pool (EA2)
	4. 2. Server (EA3)
4. Dynamischer Lookup (EA 5)	5. Dynamischer Lookup (EA 5)

Antwortzeit Mediastore

- Benutzungsprofil 1
- SPE-ED: Mittelwert 1.3317 s
- Palladio Mittelwert 1.3456 s



Ranking Mediastore Einbenutzerfall

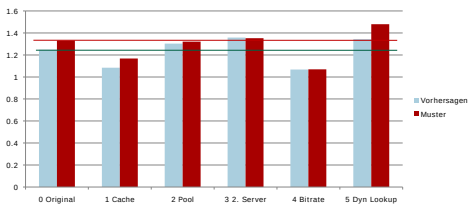
SPE-ED	Palladio
1. Caching (EA 1)	1. Bitrate senken (EA 4)
2. Bitrate senken (EA 4)	2. Caching (EA 1)
3. DBC Pool (EA2)	3. DBC Pool (EA2)
4. 2. Server (EA3)	4. 2. Server (EA3)

Unterschiede zur intuitiven Abschätzung:

- Favorit bei Palladio anders
- Reihenfolge für DBC Pool und 2. Server

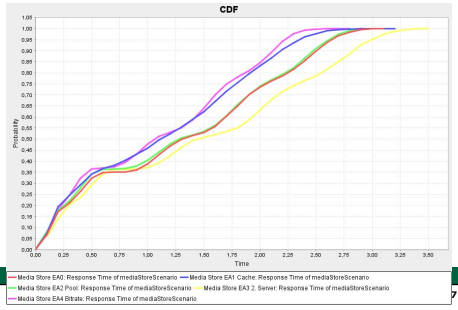
Mediastore Vorhersagen SPE-ED

- Vorhersagen mit SPE-ED
- Vergleich zur „Musterlösung"



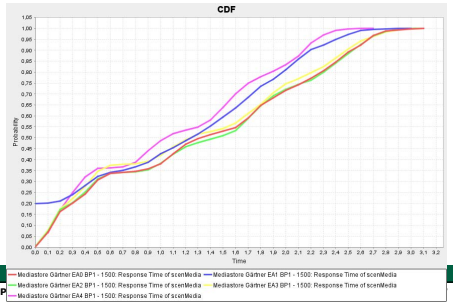
Mediastore Vorhersagen Palladio

- Entwurfsalternativen BP1



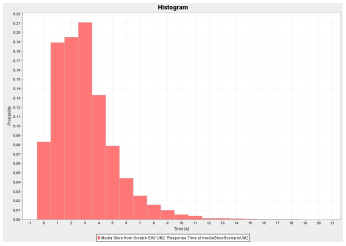
Mediastore Vorhersagen Palladio

- Entwurfsalternativen BP1



Antwortzeit Mediastore

- Benutzungsprofil 2
- SPE-ED: Mittelwert 8.9139 s
- Palladio Mittelwert 2.9739 s



Ranking Mediastore Mehrbenutzerfall

SPE-ED

1. Caching (EA 1)
2. Bitrate senken (EA 4)
3. DBC Pool (EA2)
4. 2. Server (EA3)

Palladio

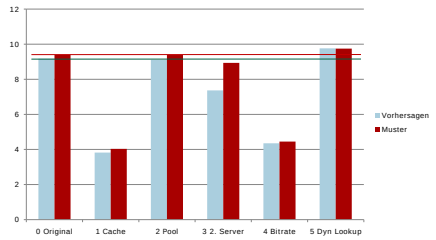
1. Bitrate senken (EA 4) und Caching (EA 1)
2. DBC Pool (EA2)
3. 2. Server (EA3)

Unterschied zur intuitiven Abschätzung:

- Bitrate nicht Favorit

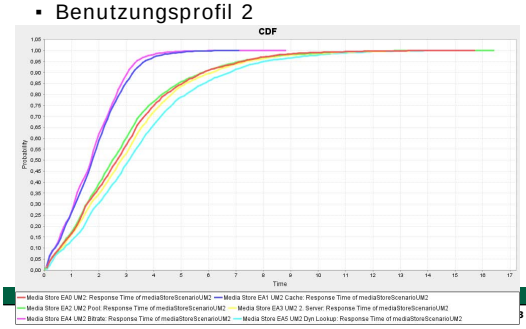
Mediastore Vorhersagen SPE-ED

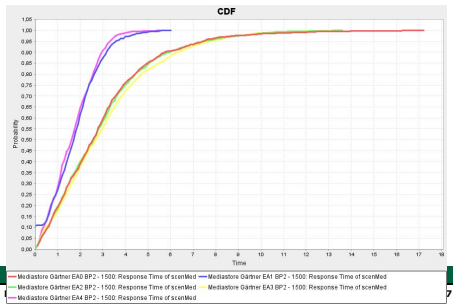
- Benutzungsprofil 2



Mediastore Vorhersagen Palladio

- Benutzungsprofil 2





Offen: Antwortzeit des wirklichen System

- Für die Bewertung der vorhergesagten Werte
- Prototypische Implementierung des Systems
- Messungen mit gegebenen Benutzungsprofilen

Abschätzung Webserver

Benutzungsprofil 1 (1 Benutzer)

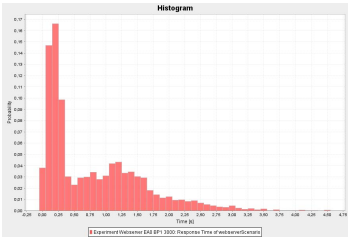
1. Caching (EA 1)
2. 2. Server(EA 4)
3. Logging (EA3)
4. Threadpool (EA5)
5. Dynamischer Lookup (EA 2)

Benutzungsprofil 2 (mehrere Benutzer)

1. Caching (EA 1)
2. 2. Server(EA 4)
3. Threadpool (EA5)
4. Logging (EA3)
5. Dynamischer Lookup (EA 2)

Antwortzeit Webserver

- Benutzungsprofil 1
- SPE-ED: Mittelwert 0.7599 s
- Palladio Mittelwert 0.8086 s



Ranking Webserver Einbenutzerfall

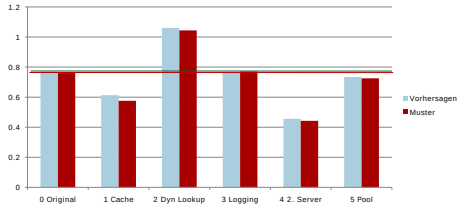
SPE-ED	Palladio
1. 2. Server(EA 4)	1. 2. Server(EA 4)
2. Caching (EA 1)	2. Caching (EA 1)
3. Threadpool (EA5)	3. Logging (EA3)
4. Logging (EA3)	4. Threadpool (EA5)

Unterschiede zur intuitiven Abschätzung:

- 2. Server als beste Alternative
- Einordnung Threadpool bei SPE-ED anders

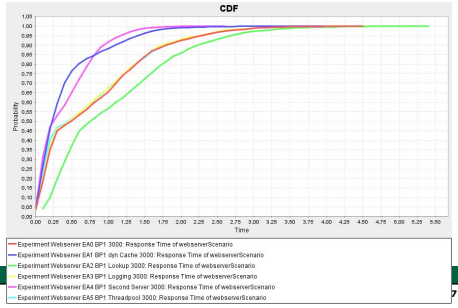
Webserver Vorhersagen SPE-ED

- Vorhersagen mit SPE-ED
- Vergleich zur „Musterlösung"



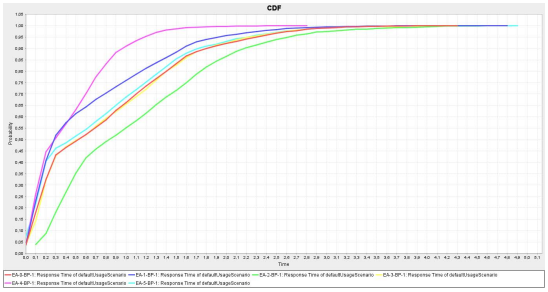
Webserver Vorhersagen Palladio

- Entwurfsalternativen BP1



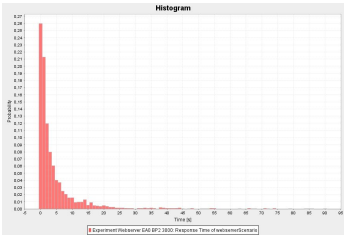
Webserver Vorhersagen Palladio

- Entwurfsalternativen BP1



Antwortzeit Webserver

- Benutzungsprofil 2
- SPE-ED: Mittelwert 1.3941 s
- Palladio Mittelwert 4.6195 s



Ranking Webserver Mehrbenutzerfall

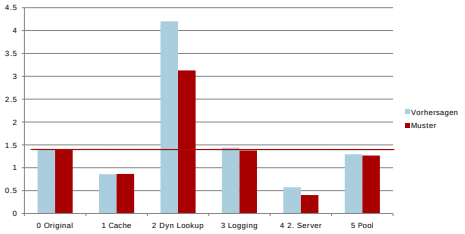
SPE-ED	Palladio
1. 2. Server(EA 4)	1. 2. Server(EA 4)
2. Caching (EA 1)	2. Caching (EA 1)
3. Threadpool (EA5)	3. Threadpool (EA5)
4. Logging (EA3)	4. Logging (EA3)

Unterschiede zur intuitiven Abschätzung:

- 2. Server als beste Alternative

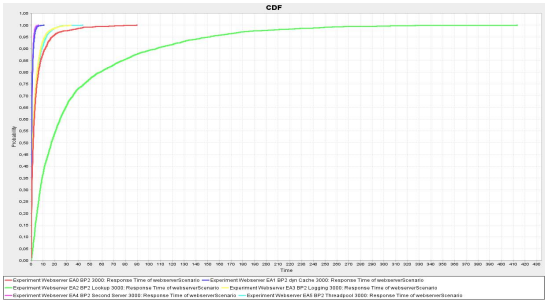
Webserver Vorhersagen SPE-ED

- Benutzungsprofil 2



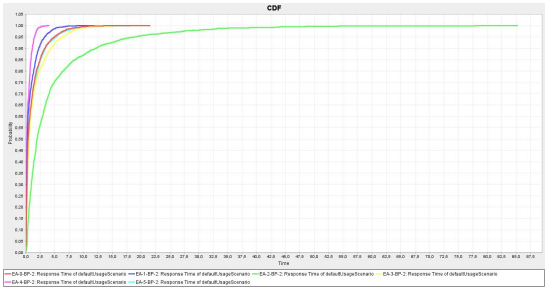
Webserver Vorhersagen Palladio

- Entwurfsalternativen BP2



Webserver Vorhersagen Palladio

- Entwurfsalternativen BP2



Ingenieurmäßiger Software-Entwurf

„Lessons Learned" / Rückblick

Was Sie gelernt haben (1)

- Die Abschätzung der Performanz selbst kleinerer Software-Systeme ist nicht trivial und fehlerträchtig
- Sie haben verschiedene Entwurfsalternativen kennen gelernt
- Sie kennen wichtige Einflussfaktoren auf die Performanz

Was Sie gelernt haben (2)

- Sie haben zwei Prozesse und Verfahren zur systematischen Performanz-Bewertung kennen gelernt
  - SPE-ED: „State of the Art"
  - Palladio: Unterstützt die Besonderheiten von Komponenten
- Andere Verfahren (bspw. KLAPER) lassen sich damit leicht von Ihnen anwenden: Konzepte und Ideen sind bekannt

Was Sie gelernt haben (3)

- Für eine systematische Wahl von Entwurfsalternativen (unter Berücksichtigung von QoS-Aspekten)
  - Reicht Intuition nicht aus
  - Reichen verbreitete Software- / Komponentenmodelle nicht aus
  - Ist eine koordinierte Zusammenarbeit verschiedener Entwicklerrollen erforderlich
  - Sind Spezialisten erforderlich, die weitergehende Fähigkeiten als ein Programmierer haben

Was Sie gelernt haben (4)

- Genau spezifizierte Komponenten erlauben die zielgerichtete Wahl von Entwurfsalternativen
  - Fordert von Komponentenentwicklern eine Modellierung
  - Entsprechen der Praxis von Ingenieuren

R=2,3 Ω
V_{max}=...
...

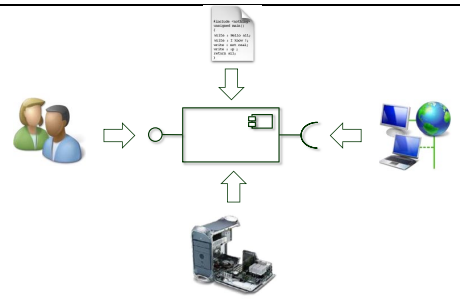
  - Über Aussagen auf Modellebene kann das bestehende „Trial-and-Error"-Verfahren abgelöst werden

Bildquelle: pixelio.de

Was Sie gelernt haben (5)

- Die Implementierung einer Komponente folgt vertraglich zugesicherten Eigenschaften
  - Bindend für Komponentenimplementierer
  - Verlässlich für Komponentenverwender
- SEFFs entsprechend einem erweiterten Vertragsmodell

Komponenten Performanz



[Becker2006a]

Einflussfaktoren (1)

- Parametrisierung über Einflussfaktoren muss explizit gemacht werden
  - Benutzungsprofil
  - Externe Dienste
  - Ausführungsumgebung
  - Implementierung von Komponentendiensten

Einflussfaktoren (2)

- Interne Aktionen von Komponenten können zu einem einzelnen Modellkonstrukt zusammengefasst werden
- Schleifendurchläufe und Verzweigungen entscheiden über die Performanz, wenn
  - Externen Dienstaufrufe auftreten
  - Parametrische Abhängigkeiten existieren
- Komponenten-Performanz lässt sich nicht durch fixe Zeitwerte beschreiben

Komponenten

- Ein Komponentenmodell für Performanzvorhersage
  - Umfasst mehr als nur angebotenen und benötigte Schnittstellen
  - Kann auf einem ausgereiften Schnittstellenmodell inklusive Protokollen aufbauen
  - Enthält zwingend eine Entsprechung für SEFFs

System

- Entwurfsentscheidungen auf der Architekturebene erfolgen zumeist auf Systemniveau
- Alternativ stellen auch Composite-Components „Mini-Systeme" mit „Mini-Architektur" dar, die Entwurfsentscheidungen bedürfen
- Auf der Systemebene ist zwischen Wiederverwendung und Neuentwicklung von Komponenten abzuwägen

Ressourcen

- Eine Unterscheidung zwischen verschiedenen Ressourcen ist notwendig
  - CPU / Speicher / Festplatte
  - Semaphore / Threads / Datenbankverbindungen
- Je nach Situation kann die Einführung eigener (genauerer) Ressourcen sinnvoll sein
  - Lesende / schreibende Zugriffe
  - Art der CPU-Berechnungen (Integer / Float)
  - ...

Ausführungsumgebung

- Detaillierte Ressourcenmodelle machen auch detaillierte Modelle der Ausführungsumgebung notwendig
- Änderungen am Modell der Ausführungsumgebung erlauben es „Sizing" und „Relocation" Fragestellungen zu beantworten

Modell ≠ Quellcode

- Implementierung ist ein *Einflussparameter* für die Komponentenperformanz
- Zu einem Modell können *n* Implementierungen existieren
- Modelle abstrahieren üblicherweise Implementierungsdetails
  - SEFF schreibt nur performanzkritische Eigenschaften einer Implementierung vor
  - Es sind beliebige (schlechte) Implementierungen denkbar, die nicht dem modellierten SEFF entsprechen
    → keine Performanzvorhersagen möglich

Modellgetriebene Entwicklung

- Produktivitätszuwächse sind möglich, wenn Performanz-Modelle nicht weggeworfen werden
  - Durch Quellcode-Generierung wird Entwicklungsaufwand gespart
  - Höhere Konformität zwischen Modell und Code
  - → Bessere Vorhersagbarkeit von Eigenschaften

Status

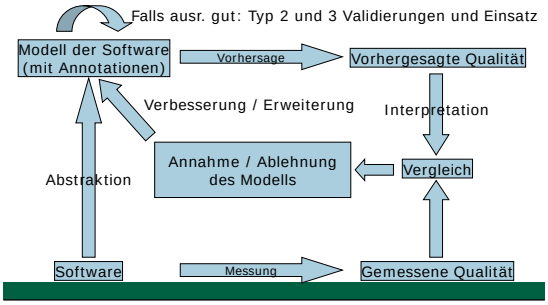
- Software-Entwicklung steht am Anfang, eine Ingenieursdisziplin zu werden
- Teil einer weiteren Spezialisierung in der Software-Entwicklung
- Bewusstsein der Notwendigkeit systematischer ingenieurmäßiger Software-Entwicklung in der Industrie nimmt stark zu

Weitere Validationen und Einsatz

- Typ 1: Validation des Vorhersagemodells
- Typ 2: Validation der Einsetzbarkeit
  - Fallstudien und kontrollierte Experimente mit Studierenden
- Typ 3: Validation des Nutzens
  - im Gegensatz zu anderen Methoden
  - Grenzen des Ansatzes
  - Notwendige Voraussetzungen
  - FZI
  - Partner

FZI

Wissenschaftliches Vorgehen: Typ 1 Validation



Zusammenfassung

- Abschätzung der Auswirkung von Entwurfsentscheidungen ist Kennzeichen einer Ingenieurdisziplin
- Systematische Behandlung von Software-Qualitätseigenschaften erfordert analytische Vorhersagemodelle
  - nahe an Software-Architektur (Erzeugung / Interpretation)
- Modellbildungsprozess durch naturwissenschaftliches Vorgehen
  - Kein Beweisbegriff, stattdessen Validierungen erforderlich

Software-Entwicklung wird modell-zentrierter.
Code-Zentrierung ist so sinnvoll wie
"**Lötzinn-Zentrierung**" eines Elektrotechnikers

Fragebögen

Zeit

- Jeweils 15 Minuten pro Fragebogen

A.5 Preparatory Exercises

The first preparatory exercise reviews basics of component-based software engineering, and was created by Klaus Krogmann. 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 (cf. section 4.1.2 for details of the exercises). For the exercises 3,4,5 and 6, I also created sample solution sheets. All sample solutions were discussed in the lab session.

Ausgabe: 25.04.2007

Abgabe der Lösungen: 01.05.2007

Vorstellung der Musterlösung: 04.05.2007

Übungsblatt 1

Ziel der Übung: Grundlegendes Verständnis von Komponenten, komponentenbasierter Software-Entwicklung, dem Prozess komponentenbasierter Software-Entwicklung sowie den Rollen innerhalb des Prozesses.

Allgemeines

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an krogmann@ipd.uka.de.

Abgabe der Lösungen

Alle Übungsblätter werden in Zweier-Gruppen bearbeitet. Die Einteilung der Gruppen erfolgt im Praktikum und wechselt (die Einteilung geben wir jeweils rechtzeitig bekannt). Bitte vermerken Sie auf Ihren Abgaben stets ihre Namen und Matrikelnummern. Die abgegebene Lösung legen Sie jeweils bitte in das SVN *beider* Gruppenmitglieder – es gibt also zwei identische elektronische Abgabeversionen.

Legen Sie im SVN-Repository bitte pro Übungsblatt ein Verzeichnis an, in das Sie gesammelt die zu einer Lösung gehörenden Dateien ablegen. Die Verzeichnisse benennen Sie Uebung01, Uebung02, usw. – jeweils pro Übungsblatt.

Für Textdokumente akzeptieren wir die folgenden Formate:

- PDF-Dokumente
- Word-Dokumente (doc)
- OpenOffice-Dokumente
- TXT-Datei

Die Lösung muss jeweils bis zum unter „Abgabe der Lösungen“ genannten Datum um 23:59 Uhr im SVN-Repository eingecheckt werden. Unvollständige oder zu spät abgegebene Lösungen werden als nicht abgegeben gewertet. Als unvollständige Lösung gelten auch Lösungen, bei denen Teilfragen nicht beantwortet werden.

Während des gesamten Praktikums darf max. **ein** Übungsblatt als nicht abgegeben gewertet worden sein, damit das Praktikum noch als erfolgreich bestanden gewertet werden kann. Bitte bedenken Sie bei Ihren Abgaben, dass die Übungsblätter in die Praktikumsnote einfließen. Außerdem werden Kenntnisse, die Sie beim Bearbeiten der Übungsblätter erlangen, in benoteten Kurztests abgefragt. Die genauen Termine für die Kurztests entnehmen Sie bitte der Terminübersicht.

SVN-Repository

Der Pfad für den Zugriff auf das SVN-Repository lautet:

svn://[name]@141.3.52.13:/ise-[name]/

wobei [name] Ihr Benutzername ist. Der Benutzername und das dazugehörige Passwort werden Ihnen per E-Mail an die bei der Anmeldung angegeben E-Mail-Adresse mitgeteilt.

SVN-Clients

Es gibt eine große Zahl von Subversion Clients, sowie auch in Eclipse als Plugin integrierte Clients: http://subclipse.tigris.org/. Wir empfehlen Ihnen den Client von Tigris: http://tortoisesvn.tigris.org/, der sich bspw. unter Windows in den Explorer einhängt.

Aufgaben

Die Aufgaben dieses ersten Übungsblattes sind als textuelle Lösung abzugeben. Ihre Lösung sollte pro Teil-Frage (■/○) mindestens 5 Sätze enthalten.

- Begründen Sie, welche (der in der Vorlesung vorgestellten) Definitionen für eine Software-Komponente Sie bevorzugen. Gibt es eine andere Definition, die Sie favorisieren?
- Erklären Sie in eigenen Worten die in der Vorlesung vorgestellten Komponententypen:
  - Bedeutung der verschiedenen Komponententypen für den komponentenbasierten Software-Entwicklungsprozess.
  - Erörtern Sie die Vor- und Nachteile der Unterteilung in verschiedene Komponententypen
- Zeigen Sie den Unterschied zwischen Komponententyp und Komponenteninstanz auf.
- Welche Probleme könnten sich bei der Modellierung von Laufzeit-Komponenteninstanzen ergeben? Was müsste bei der Modellierung von Laufzeit-Komponenteninstanzen beachtet werden? (Folie: „Term: Component“, VL 2)
- Auf der Folie „Hierarchy of Interface Models“ (VL 2) werden die ersten drei Ebenen eines Schnittstellenmodells wiedergegeben. Welche Ebene(n) könnte(n) in dem mit „…“ markierten Bereich ergänzt werden?
- Erklären Sie mit eigenen Worten die Bedeutung des Komponenten-Kontextes (VL 2).

Ausgabe: 02.05.2007

Abgabe der Lösungen: 08.05.2007

Vorstellung der Musterlösung: 11.05.2007

Übungsblatt 2

Ziel der Übung: Erste Erfahrungen mit den Performance-Vorhersagewerkzeugen SPE-ED und PCM Bench.

Allgemeines

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an krogmann@ipd.uka.de und martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wieviel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Werkzeuge

Für die Bearbeitung der Aufgaben benötigen Sie das SPE-ED Werkzeug sowie Eclipse mit der PCM Bench.

SPE-ED ist aus dem Wiki verlinkt. Das Werkzeug ist nur für Windows Rechner verfügbar. Laden Sie das Zip-Archiv herunter und packen Sie es aus, am besten direkt in Ihre SVN Repositories (Sie müssen später die meisten Dateien einchecken). Bevor Sie SPE-ED starten können, müssen Sie die Systemzeit Ihres Rechners auf das Jahr 2003 zurück stellen. Starten Sie SPE-ED durch Ausführung von speed3B.exe. Während der Ausführung des Werkzeugs kann die Systemzeit bereits wieder berichtigt werden.

Eine Version von Eclipse mit allen für die PCM benötigten externen Plugins ist ebenfalls aus dem Wiki verlinkt. Sie können diese parallel zu einer bereits bestehenden Eclipse Installation verwenden. Richten Sie dann aber einen neuen Workspace ein. Nach dem ersten Starten von Eclipse müssen Sie die aktuellsten Versionen der PCM Plugins herunterladen. Wählen Sie dazu Hilfe->Software updates->Find and Install aus und fügen Sie dort die PCM Stud Updatesite als Remote Site hinzu: `http://sdqweb.ipd.uka.de/eclipse/PCM_stud`
Aktualisieren Sie alle verfügbaren Plugins. Laden Sie weiterhin das im Wiki verlinkte Repository mit vorbereiteten primitiven Datentypen (PrimitiveTypes.repository) herunter.

Aufgaben

1. SPE-ED: Arbeiten Sie die erste Demo aus dem Quickstart Guide des SPE-ED Handbuchs durch (Seiten 4 bis 25) und erstellen Sie das Projekt wie angegeben. Legen Sie das gesamte SPE-ED Verzeichnis ohne die .exe Datei in Ihre SVN Repositories ab.

Informationen zum Hintergrund von SPE-ED können Sie ebenfalls im Handbuch finden, sie werden aber auch in späteren Vorlesungen noch vorgestellt werden.

2. PCM Bench: Legen Sie ein neues Palladio Projekt an und erstellen Sie einige Inhalte. Gehen Sie dazu wie unten angegeben vor. Lassen Sie sich vom Umfang der Erläuterungen nicht abschrecken, sie sind sehr detailliert und daher lang. Der tatsächliche Aufwand zur Umsetzung ist jedoch nicht so hoch, wie befürchtet werden könnte.

   Im Wiki finden Sie weiterhin ein Tutorial in Form eines Screencasts, der die hier beschriebenen Aktionen nochmal in ähnlicher Form zeigt. Folgen Sie jedoch nicht nur dem Screencast, es bestehen einige kleine Unterschiede. Ausschlaggebend für die Bewertung sind die unten angegebenen Aufgaben. Weitere Informationen zum Hintergrund des Palladio Werkzeugs werden in späteren Vorlesungen noch vorgestellt werden.

   a) Erstellen Sie ein neues Projekt über File -> new Project, wählen Sie dort General -> Project aus. Geben Sie im Projektnamen unbedingt Ihre Übungsgruppennummer an!

   b) Legen Sie ein Repository an. Wählen Sie dazu im Kontextmenü des neuen Projektes (Rechtklick auf das Projekt) New -> Other aus. Wählen Sie in der folgenden Ansicht Example EMF Model Creation Wizards -> Repository Model aus. Stellen Sie im folgenden Dialog sicher, dass das Repository in dem neuen Projekt angelegt wird. Im nächsten Dialog wählen Sie als Model „Repository“ aus. Klicken Sie „Finish“. Sie erhalten eine Repository Datei, darin befindet sich das Repository Modell, dass zunächst noch „aName“ heißt.

   c) Benennen Sie das Repository Modell in repository1 um. Wählen Sie dazu die neu angelegte Repository Datei im Explorer (links) aus und öffnen Sie dazu in der Resource Set Ansicht in der Mitte die Baumstruktur. Verwenden Sie den „Properties“ Reiter unten, um das Repository umzubenennen.

   d) Legen Sie ein neues Diagramm zu diesem Repository an. Sie müssen dazu die Project Explorer Ansicht links haben. Im Kontextmenü der Repository Datei wählen Sie „Initialize repository_diagram diagram file“ aus. Stellen Sie im folgenden Dialog sicher, dass das Repository Diagram in dem richtigen Projekt angelegt wird. Im nächsten Dialog wählen Sie als Basis für das Diagramm das bereits angelegte Repository Modell aus. Klicken Sie „Finish“.

   e) Um im Projekt primitive Datentypen verwenden zu können, laden Sie das vorbereitete Repository mit primitiven Datentypen in Ihr Repository. Öffnen Sie dazu das Repository oder das Repository Diagram in der mittleren Ansicht und wählen Sie im Kontextmenü „Load Resource“ aus. Wählen Sie „Browse Filesystem“ und navigieren Sie zu dem heruntergeladenen PrimitiveTypes.repository. Wählen Sie dieses aus.

   f) Legen Sie im Repository zwei Basic Components und drei Interfaces an, dies können Sie im Repository Diagramm über die Palette oder über den Baumeditor des Repository Modells. Benennen Sie die Komponenten und Interfaces um, der Name kann beliebig gewählt werden.

   g) Geben Sie den Interfaces jeweils eine Signatur, von denen mindestens eine einen Parameter und eine einen Rückgabewert haben soll. Sie geben Rückgabewerte und Parameter für Signaturen an, indem Sie das Interface auswählen, und im Properties

Bereich unten die „Operations“ Sicht auswählen. Dort können Sie Rückgabewert und Parameter angeben.

h) Verbinden Sie jede Komponente mit jeweils zwei Interfaces, eines in der Provided-Rolle, eines in der Required-Rolle, so dass alle drei Interfaces verwendet werden und ein Interface von der einen Komponente angeboten und von der anderen benötigt wird (Sie vermuten es: Sie sollen später zusammengefügt werden). Wählen Sie hierzu aus der Palette den Required Role bzw. den Provided Role Verbinder, setzen Sie bei der zu verbindenden Komponente an und ziehen Sie die Rolle zum gewählten Interface.

i) Erstellen Sie ein System Diagram für Ihr Projekt. Wählen Sie diesmal dazu im Kontextmenü des Projekts New -> Other aus. Wählen Sie in der folgenden Ansicht Palladio Modeling -> system diagram aus. Stellen Sie im folgenden Dialog sicher, dass das Repository in dem neuen Projekt angelegt wird. Klicken Sie „Finish“. Das zugehörige System Model wird dabei automatisch erzeugt. Sie haben somit einen zweiten Weg kennengelernt, wie ein Model und ein zugehöriges Diagramm neu erzeugt werden können.

j) Öffnen Sie das System Model und definieren Sie den Assembly Context für die beiden eben erstellten Komponenten. Dazu muss zunächst Ihr Repository in das System geladen werden. Öffnen Sie dazu das Kontextmenü im System Model oder im System Diagram und wählen Sie „Load Resource“ (Klicken Sie dabei nicht auf das bereits erzeugte <<system>>). Diesmal können Sie „Browse Workspace“ auswählen. Verwenden Sie Ihr eben angelegtes Repository.

   Wählen Sie nun in der Palette des System Diagrams für jede Komponente den Assembly Context und setzen Sie Ihn in bereits generierte System (<<system>>). Im erscheinenden Dialog wählen Sie jeweils eine Komponente aus.

k) Verbinden Sie die Komponenten an den passenden Schnittstellen mit einem Assembly Connector.

l) Erzeugen Sie ein Required und ein Provided Interface für das System, die dieselbe Schnittstelle wie die entsprechende noch unverbundene Schnittstelle der Komponenten anbietet.

m) Verbinden Sie die Required und ein Provided Interfaces des Systems mit den entsprechenden freien Schnittstellen der Komponenten. Verwenden Sie dazu den RequiredDelegationConnector bzw. den ProvidedDelegationConnector.

Mit dieser Übung haben Sie noch nicht den vollen Umfang des PCM Werkzeugs kennengelernt, hierzu werden Sie in weiteren Übungen kommen. Legen Sie das resultierende Projekt in Ihre SVN Repositories ab.

3. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wieviel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:

   Aufgabe 1 SPE-ED: Zeitangabe

   Aufgabe 2 PCM Bench: Zeitangabe

Ausgabe: 09.05.2007

Abgabe der Lösungen: 15.05.2007

Vorstellung der Musterlösung: 18.05.2007

Übungsblatt 3

Ziel der Übung: Performanzanalyse mit SPE-ED zu einer frühen Phase im Entwicklungsprozess.

Allgemeines

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Werkzeuge

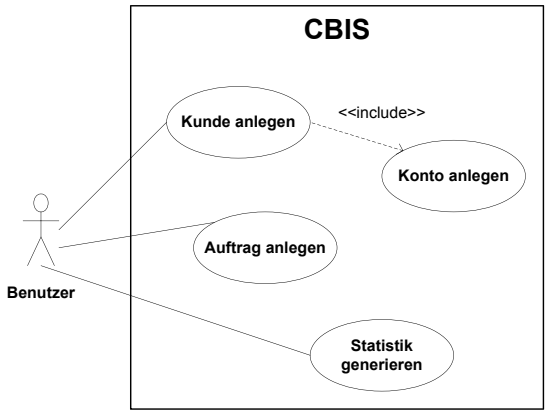
Für die Bearbeitung der Aufgaben benötigen Sie das SPE-ED Werkzeug, dass Sie in der zweiten Übung bereits verwendet haben.

Aufgaben

1. Ein neues, leichtgewichtiges Informationssystem für ein Handelsunternehmen soll komponentenbasiert entwickelt werden. Sie sind der Performanzanalyst im Entwicklungsteam, und sollen sicherstellen, dass bereits der Entwurf des Systems gute Antwortzeiten der späteren Implementierung ermöglicht.

   Das Entwicklungsteam hat bereits einige erste Anwendungsfälle, die Aufteilung in zuständige Komponenten und Szenarien für das neue „Component Based Information System“ identifiziert, die in den folgenden Diagrammen dargestellt sind. Da die bestehenden Datenbanken des Unternehmens genutzt werden sollen, werden diese als systemextern modelliert.

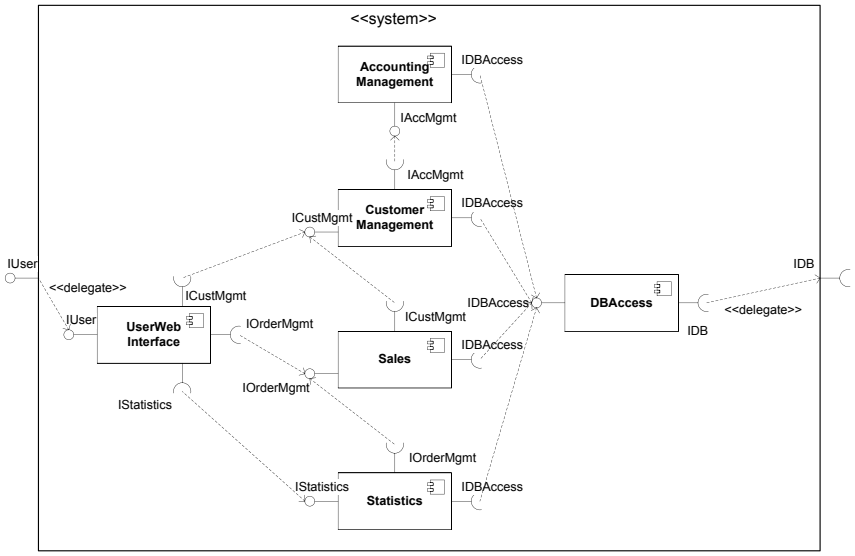
Anwendungsfälle:



Es wurden drei Anwendungsfälle identifiziert: Das Anlegen eines Kunden, dass das Anlegen eines neuen Kontos im Rechnungswesen mit einschließt, das Anlegen eines neuen Auftrags mit verschiedenen Auftragspositionen und das Generieren einer Statistik über bisherige Aufträge.

Entwurf der Architektur:

Die einzelnen Funktionsbereiche werden in Komponenten realisiert: CustomerManagement, AccountingManagement, Sales und Statistics. Zusätzlich ist die Komponente DBAccess für den Datenbankzugriff zuständig, sie abstrahiert von den SQL Anfragen an die Datenbank und stellt dem restlichen System eine objektorientierte Schnittstelle bereit. Die Komponente UserWebInterface wandelt die Anfragen, die über HTTP eingehen, in Anfragen an die einzelnen Funktionskomponenten um.



Szenarien

Für die drei Anwendungsfälle wurden Sequenzdiagramme erstellt. Das Verwalten von Sessions, um die einzelnen HTTP Anfragen einer Transaktion zuordnen zu können, wurde noch nicht modelliert.

Szenario Kunde anlegen:

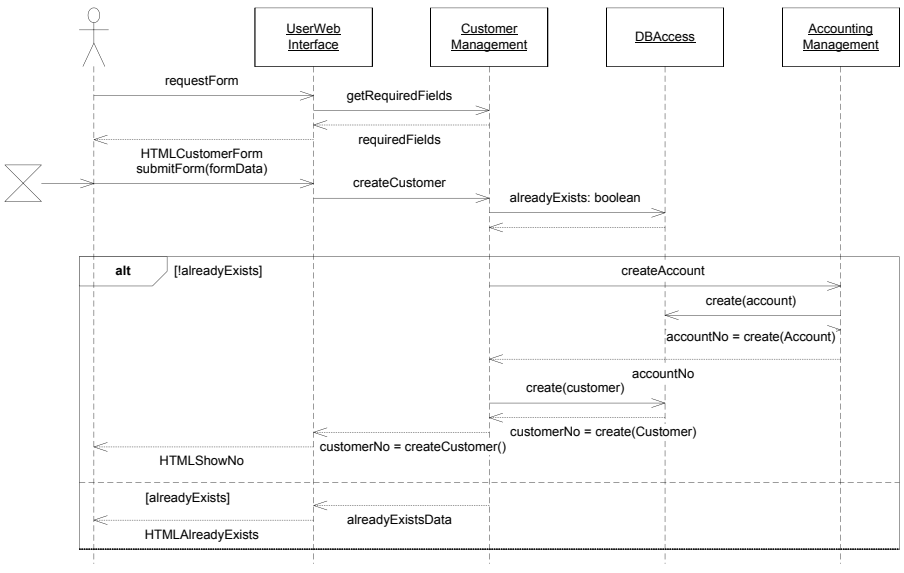
Der Benutzer wählt aus, dass ein Kunde angelegt werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für das Anlegen eines Kunden benötigten Daten beim CustomerManagement ab. Danach wird die HTML Seite mit den entsprechenden Formularen für die Kundendaten generiert und zurückgeschickt.

Der Benutzer füllt das Formular aus, was einige Sekunden dauert (durch die Sanduhr dargestellt). Danach wird die Seite wieder an das UserWebInterface geschickt, dort werden die Daten aus dem HTTP Request extrahiert und die CustomerManagement Komponente aufgerufen. Diese prüft zunächst, ob der eingegebene Kunde nicht bereits vorliegt.

Falls nicht, wird zunächst ein neues Konto im Rechnungswesen für diesen Kunden angelegt. Danach wird der Kunde selbst in die Datenbank gespeichert. Die Datenbank erzeugt eine Kundennummer, die zurückgegeben wird.

Falls der Kunde bereits vorliegt, wird dem Benutzer eine entsprechende Meldung dargestellt. Es wird angenommen, dass nur in 5% der Fälle der Kunde bereits erfasst wurde.

Kunde anlegen



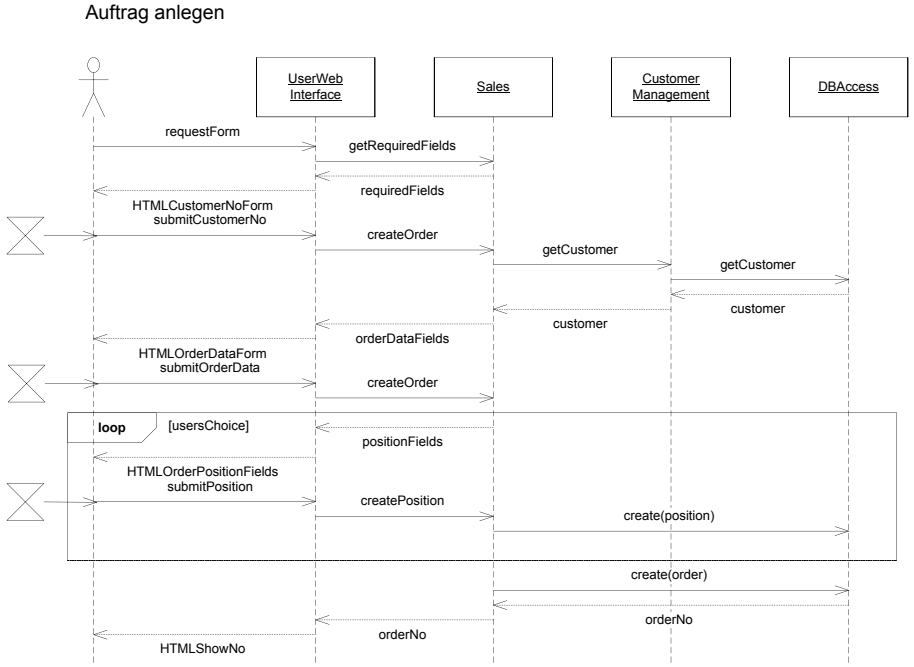
Szenario Auftrag anlegen:

Auch hier wählt der Benutzer aus, dass ein Auftrag angelegt werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für das Anlegen eines Kunden benötigten Daten bei der Sales Komponente ab. Danach wird die HTML Seite mit den entsprechenden Formularen, hier die Eingabe einer Kundennummer, generiert und zurückgeschickt.

Der Benutzer gibt die Kundennummer an, danach wird die Seite wieder an das UserWebInterface geschickt, dort werden die Daten aus dem HTTP Request extrahiert und die Sales Komponente aufgerufen. Diese holt zur angegeben Kundennummer den entsprechenden Kunden aus der Datenbank, über die DBAccess Komponente.

Der Benutzer wird nun aufgefordert, einige Kopfdaten zum Auftrag wie Lieferdatum etc. einzugeben, auch hier vergehen einige Sekunden. Nachdem die Daten an das System geschickt wurden, werden nun die einzelnen Positionen beim Benutzer abgefragt und gleich in der Datenbank gespeichert. Ein typischer Auftrag umfasst 10 Positionen. Der Benutzer kann bei der Eingabe der Positionen entscheiden, ob er eine weitere Position hinzufügen möchte (usersChoice enthält diese Information im Diagramm).

Nachdem alle Positionen erfasst wurden, wird auch der Auftrag selbst in der Datenbank abgelegt. Dem Benutzer wird eine Seite mit der generierten Auftragsnummer angezeigt.



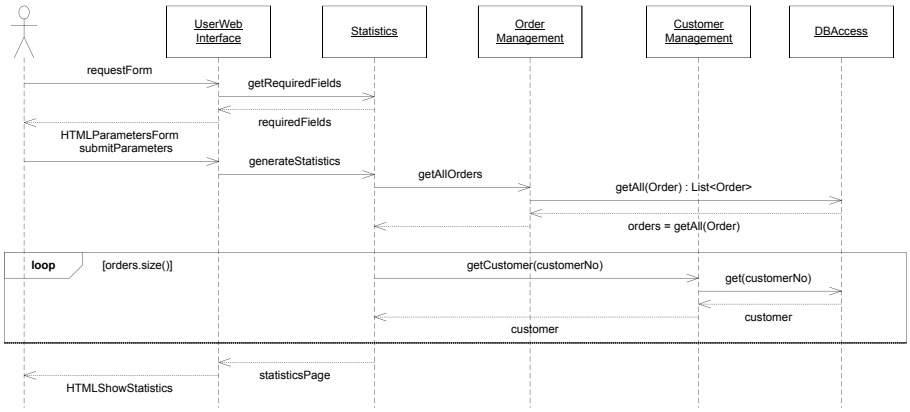
Szenario Statistik generieren

Das System soll Statistiken zu Kunden und deren Aufträgen generieren können. Hier sind verschiedene Auswertungen denkbar: Es könnte der beste Kunde, d.h. der Kunde mit dem höchsten Auftragsvolumen, ermittelt werden, oder auch verschiede Diagramme, die Aufträge und/oder Kunden graphisch aufbereiten. Dabei sollen auch Daten von Kunden angezeigt werden, die der Auftrag nicht vorhält, die Kunden sollen daher einzeln geladen werden.

Zunächst wählt der Benutzer aus, dass eine Statistik generiert werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für die Statistikgenerierung benötigten Daten wie Art der Statistik und Zeitraum bei der Sales Komponente ab. Danach wird die HTML Seite mit den entsprechenden Formularen generiert und zurückgeschickt.

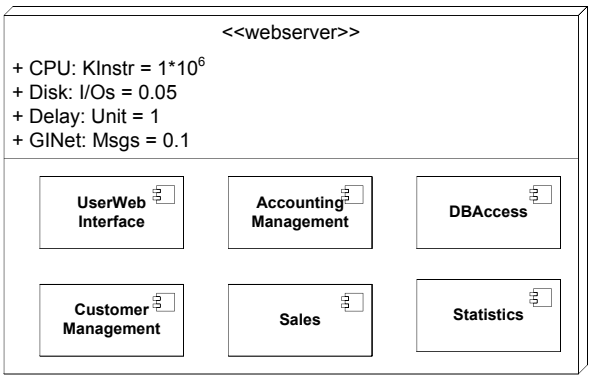
Um die Statistik zu generieren, lädt die Statistics Komponente nun alle Aufträge, und für jeden Auftrag den entsprechenden Kunden aus der Datenbank, über diezuständige Komponente OrderManagement bzw. CustomerManagement. Schließlich wird die angeforderte Statistik berechnet und die Seite mit den Ergebnissen generiert und zurückgeliefert.

Statistik generieren



Gegebene Daten:

- Das Software Resource Template enthält die Software Resources: Work Units, DBAccess, UserDelay, Message
- Die Komponenten sind auf folgendem Webserver eingesetzt:



- Verwenden Sie die mit SPE-ED mitgelieferte Facility Webserver (bei der Auswahl "Show all" auswählen, dort die Facility Webserver).
- Die folgende Overhead Matrix kann angenommen werden:

Devices	CPU	Disk	Delay	GINet
Quantity	1	1	1	1
Service Units	KInstr	I/Os	Units	Msgs
WorkUnit	10			
DBAccess	50	3		
UserDelay			1	
Message	30			1
Service time	1e-006	0.05	1	0.1

- Angaben zu den Ressourcenanforderungen:
  - Das Parsen der Anfragen dauert etwas länger: 5 WorkUnits
  - Das Parsen der vom Benutzer ausgefüllten Formulare benötigt 10 Work Units.
  - Das Generieren der Seiten vor dem Zurückschicken benötigt 10 WorkUnits
  - Das Anlegen eines Datensatzes benötigt 5 DBAccess
  - Einfache Abfragen der Datenbank (z.B. Exists) benötigen 1 DBAccess
  - Die Lieferung von Ergebnissen der Datenbank benötigt 1 DBAccess pro 100 Ergebnisse, mind. jedoch 1.

- Jeder Aufruf einer anderen Komponente benötigt 1 WorkUnit durch den Overhead des Aufrufs.
- Jede Fallunterscheidung benötigt 1 WorkUnit.
- Benutzer benötigen im Mittel ca. 10 Sekunden um Daten einzugeben.
- Pro Bestellung 5 Work Units

- Es sind 12000 Aufträge im System erfasst bei 5500 Kundendatensätzen.

Performanzziel

Die einzelnen Szenarien sollen nicht länger als 20 Sekunden dauern.

Ihre Aufgaben

a. Legen Sie für diese drei Szenarien Software Execution Models an. Sie können einige Operationen zusammenfassen, wo dies sinnvoll ist. Fassen Sie aber nur zusammen, wenn die Operationen auch einen logischen Zusammenhang haben, damit die Verständlichkeit gegeben ist, und geben Sie der entstehenden Node einen sprechenden Namen, der die Zusammenfassung verdeutlicht. Machen Sie weiterhin Gebrauch von der Schachtelung des Ausführungsgraphen (Expand). Denken Sie aber daran, dass nur drei Subgraphen in dieser Version des SPE-ED Werkzeugs angelegt werden können. (Nur im SPE-ED Werkzeug)

b. Lösen Sie die einzelnen Modelle und betrachten Sie die Antwortzeiten. Geben Sie für die einzelnen Szenarien an, ob das Performanzziel erreicht werden kann sowie die ermittelten Werte. Analysieren Sie, wo ggf. Probleme verursacht werden. Welche der in der VL zum SPE vorgestellten Performanz-Patterns oder -prinzipien werden verletzt? Welche Anti-Patterns werden verwendet?

c. Geben Sie mindestens drei Entwurfsalternativen an, die die Performanz der problematischsten Stellen verbessern können. Begründen Sie.

Geben Sie Ihre Lösungen zu b. und c. in einem separaten Dokument namens uebung03 ab, in einem der erlaubten Dateiformate.

Legen Sie das gesamte SPE-ED Verzeichnis ohne die .exe Datei in Ihren SVN Repositories ab. Falls Sie die SPE-ED Installation aus dem vorigen Übungsblatt verwenden wollen, können Sie die Lösung auch in das Verzeichnis zu Übung 2 hochladen. Legen Sie dann aber unbedingt im Verzeichnis „uebung03“ eine Textdatei mit einem entsprechenden Hinweis an.

Tipp: Sie können Subgraphen in anderen Szenarien wiederverwenden, indem Sie einen Knoten erweitern (expand) und, bevor Sie Knoten in den neuen Subgraph einfügen, Edit->Include… und dort den gewünschten Subgraphen auswählen.

2. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:

   Aufgabe 1 SPE-ED:

   a) Zeitangabe
   b) Zeitangabe
   c) Zeitangabe

Ausgabe: 09.05.2007

Abgabe der Lösungen: 15.05.2007

Vorstellung der Musterlösung: 18.05.2007

Übungsblatt 3 Musterlösung

Aufgaben

Szenarien

Für die drei Anwendungsfälle wurden Sequenzdiagramme erstellt. Das Verwalten von Sessions, um die einzelnen HTTP Anfragen einer Transaktion zuordnen zu können, wurde noch nicht modelliert.

Szenario Kunde anlegen:

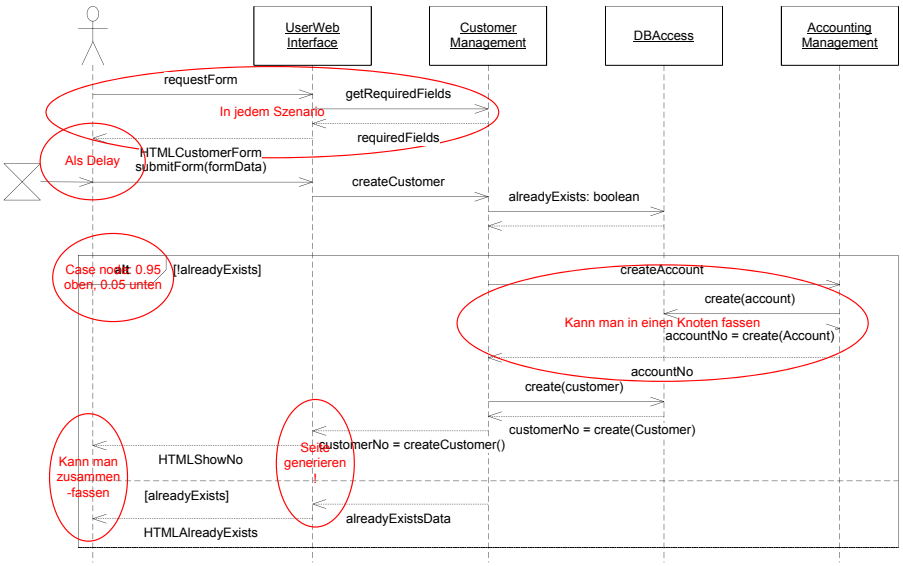
Der Benutzer wählt aus, dass ein Kunde angelegt werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für das Anlegen eines Kunden benötigten Daten beim CustomerManagement ab. Danach wird die HTML Seite mit den entsprechenden Formularen für die Kundendaten generiert und zurückgeschickt.

Der Benutzer füllt das Formular aus, was einige Sekunden dauert (durch die Sanduhr dargestellt). Danach wird die Seite wieder an das UserWebInterface geschickt, dort werden die Daten aus dem HTTP Request extrahiert und die CustomerManagement Komponente aufgerufen. Diese prüft zunächst, ob der eingegebene Kunde nicht bereits vorliegt.

Falls nicht, wird zunächst ein neues Konto im Rechnungswesen für diesen Kunden angelegt. Danach wird der Kunde selbst in die Datenbank gespeichert. Die Datenbank erzeugt eine Kundennummer, die zurückgegeben wird.

Falls der Kunde bereits vorliegt, wird dem Benutzer eine entsprechende Meldung dargestellt. Es wird angenommen, dass nur in 5% der Fälle der Kunde bereits erfasst wurde.

Kunde anlegen



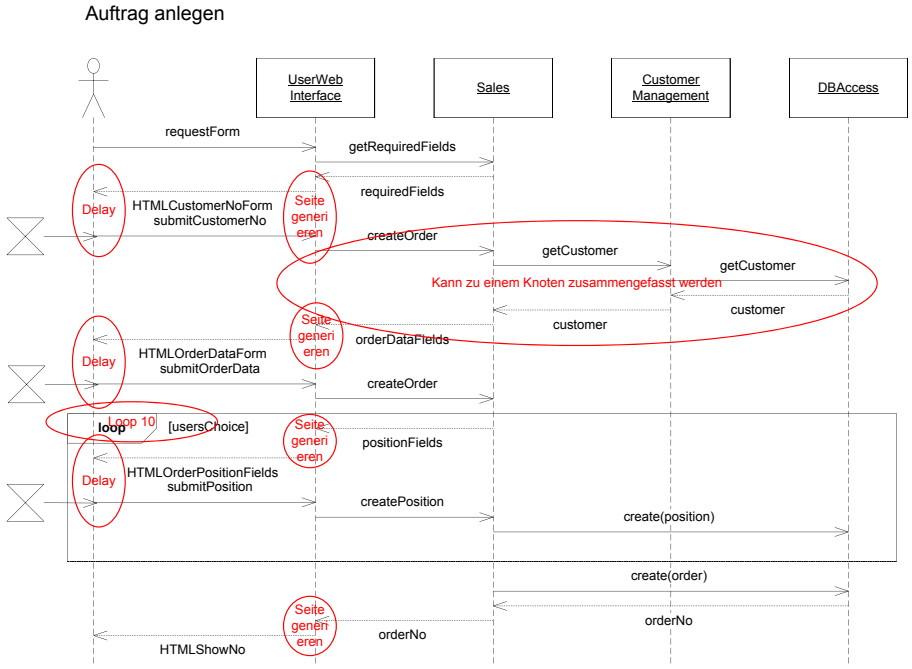
Szenario Auftrag anlegen:

Auch hier wählt der Benutzer aus, dass ein Auftrag angelegt werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für das Anlegen eines Kunden benötigten Daten bei der Sales Komponente ab. Danach wird die HTML Seite mit den entsprechenden Formularen, hier die Eingabe einer Kundennummer, generiert und zurückgeschickt.

Der Benutzer gibt die Kundennummer an, danach wird die Seite wieder an das UserWebInterface geschickt, dort werden die Daten aus dem HTTP Request extrahiert und die Sales Komponente aufgerufen. Diese holt zur angegeben Kundennummer den entsprechenden Kunden aus der Datenbank, über die DBAccess Komponente.

Der Benutzer wird nun aufgefordert, einige Kopfdaten zum Auftrag wie Lieferdatum etc. einzugeben, auch hier vergehen einige Sekunden. Nachdem die Daten an das System geschickt wurden, werden nun die einzelnen Positionen beim Benutzer abgefragt und gleich in der Datenbank gespeichert. Ein typischer Auftrag umfasst 10 Positionen. Der Benutzer kann bei der Eingabe der Positionen entscheiden, ob er eine weitere Position hinzufügen möchte (usersChoice enthält diese Information im Diagramm).

Nachdem alle Positionen erfasst wurden, wird auch der Auftrag selbst in der Datenbank abgelegt. Dem Benutzer wird eine Seite mit der generierten Auftragsnummer angezeigt.

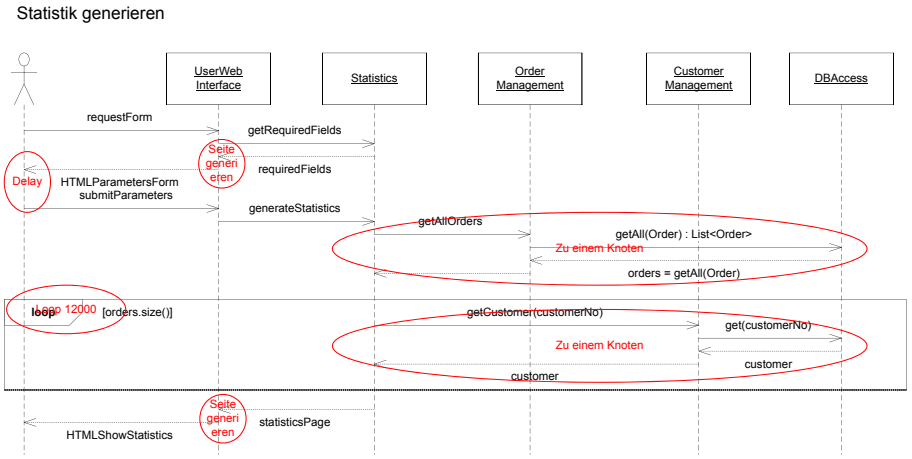


Szenario Statistik generieren

Das System soll Statistiken zu Kunden und deren Aufträgen generieren können. Hier sind verschiedene Auswertungen denkbar: Es könnte der beste Kunde, d.h. der Kunde mit dem höchsten Auftragsvolumen, ermittelt werden, oder auch verschiede Diagramme, die Aufträge und/oder Kunden graphisch aufbereiten. Dabei sollen auch Daten von Kunden angezeigt werden, die der Auftrag nicht vorhält, die Kunden sollen daher einzeln geladen werden.

Zunächst wählt der Benutzer aus, dass eine Statistik generiert werden soll, und fordert das entsprechende Formular vom UserWebInterface an. Das UserWebInterface fragt die für die Statistikgenerierung benötigten Daten wie Art der Statistik und Zeitraum bei der Sales Komponente ab. Danach wird die HTML Seite mit den entsprechenden Formularen generiert und zurückgeschickt.

Um die Statistik zu generieren, lädt die Statistics Komponente nun alle Aufträge, und für jeden Auftrag den entsprechenden Kunden aus der Datenbank, über diezuständige Komponente OrderManagement bzw. CustomerManagement. Schließlich wird die angeforderte Statistik berechnet und die Seite mit den Ergebnissen generiert und zurückgeliefert.



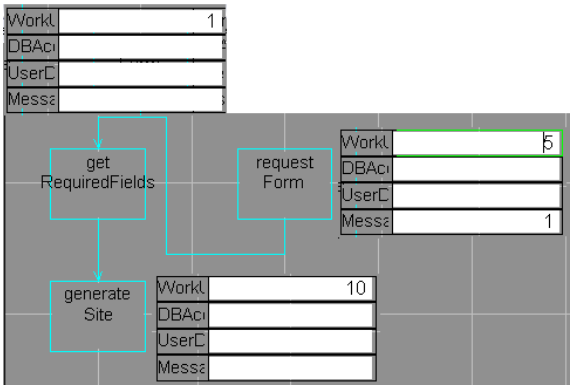
Performanzziel

Die einzelnen Szenarien sollen nicht länger als 20 Sekunden dauern.

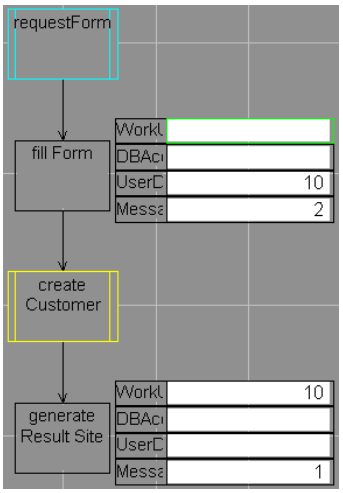
Ihre Aufgaben

a. Legen Sie für diese drei Szenarien Software Execution Models an. Sie können einige Operationen zusammenfassen, wo dies sinnvoll ist. Fassen Sie aber nur zusammen, wenn die Operationen auch einen logischen Zusammenhang haben, damit die Verständlichkeit gegeben ist, und geben Sie der entstehenden Node einen sprechenden Namen, der die Zusammenfassung verdeutlicht. Machen Sie weiterhin Gebrauch von der Schachtelung des Ausführungsgraphen (Expand). Denken Sie aber daran, dass nur drei Subgraphen in dieser Version des SPE-ED Werkzeugs angelegt werden können. (Nur im SPE-ED Werkzeug)

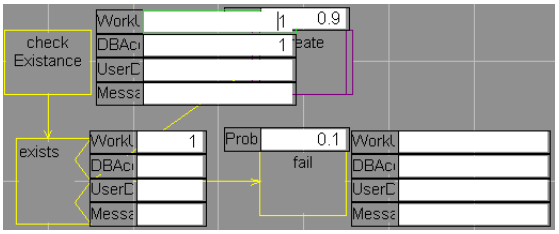
Kunde anlegen

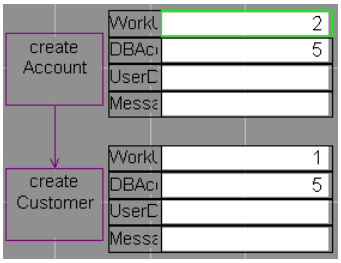


Wird unten auch wiederverwendet

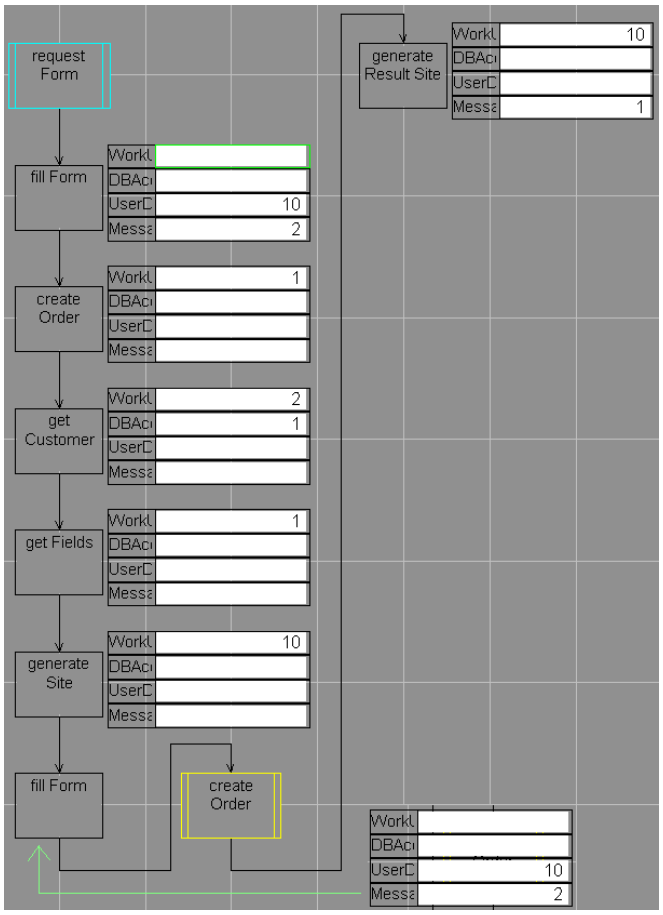


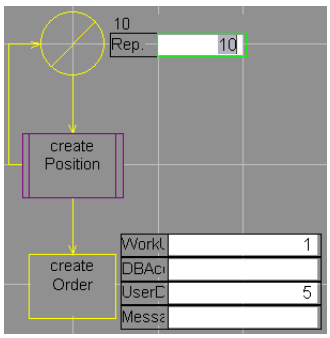
Der violette Knoten heißt „create“

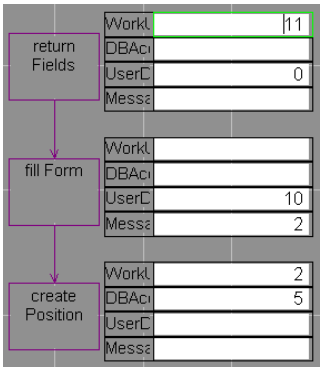




Vertrag anlegen

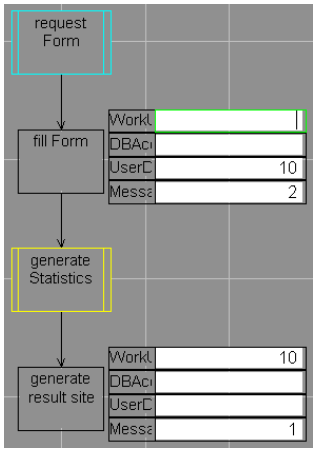


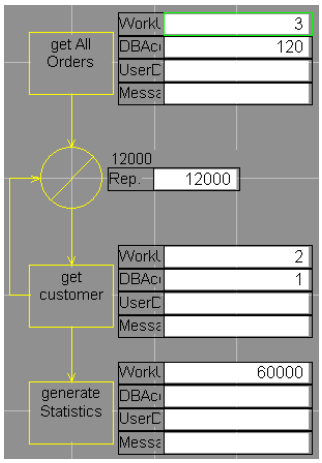




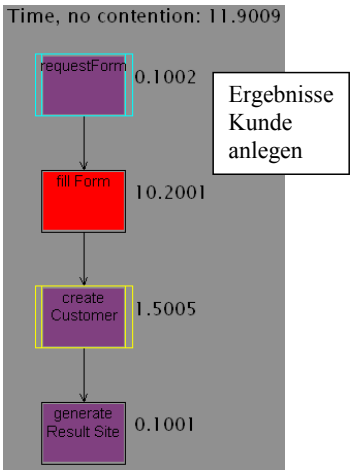
Statistik

Folgende Software Execution Graphs sind denkbar:





b. Lösen Sie die einzelnen Modelle und betrachten Sie die Antwortzeiten. Geben Sie für die einzelnen Szenarien an, ob das Performanzziel erreicht werden kann sowie die ermittelten Werte. Analysieren Sie, wo ggf. Probleme verursacht werden. Welche der in der VL zum SPE vorgestellten Performanz-Patterns oder -prinzipien werden verletzt? Welche Anti-Patterns werden verwendet?



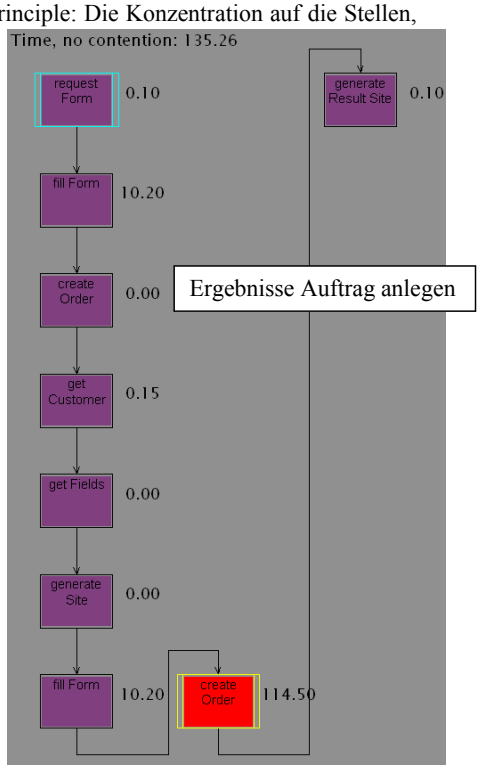
Kunde anlegen:

Es wird eine Antwortzeit von 12 Sekunden vorhergesagt. Damit kann das Performanzziel erreicht werden.

Auftrag anlegen

Es wird eine Antwortzeit von knapp 136 Sekunden vorhergesagt. Dies ist bedeutend höher als das Performanzziel. Hauptursache sind die vielen Verzögerungen durch den Benutzer, der bei dem Anlegen jeder Position 10 Sekunden lang nachdenkt. Ohne den Delay durch den Benutzer würde das Szenario noch knapp 10 Sekunden benötigen (dies kann von der Resource Usage in der Results-Sicht abgelesen werden). Es sollten also beim Anlegen der Positionen Maßnahmen getroffen werden.

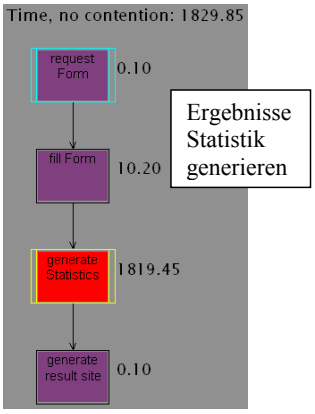
Dieser Schluss entspricht dem Centering Principle: Die Konzentration auf die Stellen, die den höchsten Workload verursachen. Um die Zeiten für das Anlegen der Positionen zu verringern, kann sich am Fast Path Pattern orientiert werden: Aufwändige Aktionen sollen beschleunigt werden. Hier ist denkbar, dem Benutzer mehrere Positionsfelder auf einer Seite anzuzeigen, so dass die Seite nicht neu geladen werden muss, und insbesondere dass der Benutzer die Seite nicht erst von neuem betrachten muss und dadurch unterbrochen wird, sondern gleich weiterarbeiten kann. Die Zeit für das Ausfüllen der Positionen würde dann wohl länger als 10 Sekunden dauern, aber wahrscheinlich weniger als 100 Sekunden für alle Positionen.



Trotzdem sollte in Erwägung gezogen werden, das Performanzziel zu verändern. Der Benutzer muss mindestens zwei Eingaben tätigen, davon wird die Eingabe der Positionen länger dauern.

Statistik generieren

Bei dem Anwendungsfall „Statistik generieren" wird das Performanzziel ebenfalls deutlich nicht erreicht. Der kritische Knoten hier ist generateStatistics, darin eingebettet der Knoten getCustomer. Die Anforderung des Knotens selbst sind gering, allerdings wird er in einer sehr häufig ausgeführten Schleife verwendet. Hier ist das Processing Versus Frequency Principle verletzt: Der relativ teure Datenbankzugriff wird zu häufig durchgeführt. Allerdings kann die Ausführung des Knoten selbst nicht günstiger gestaltet werden. Eine Option, die aber ggf. schwierig zu realisieren ist, wäre, die Schnittstelle zu DBAccess nach dem Coupling Pattern zu erweitern und eine Funktion hinzuzufügen, die gleich Bestellungen mit allen Kunden dazu zurückliefert. Hier sollte aber abgewogen werden, ob die Schnittstelle der Komponente dahingehend erweitert werden soll, oder ob dies die Wiederverwendbarkeit der Komponente beeinträchtigt. Ein Pattern, das aber in jedem Fall angewandt werden sollte, ist das Batching. Es muss nicht jeder Kunde einzeln aus der Datenbank geladen werden, dies könnte besser in Einem geschehen. Dazu müsste allerdings die Zuordnung von Kunden zu Aufträgen nachträglich geschehen. Ein weiterer Vorteil wäre, dass ein Kunde mit mehreren Aufträgen nicht mehrmals geladen werden würde.



Man könnte hier weiterhin das Antipattern Circuitous Treasure Hunt identifizieren, dass beinhaltet, dass ein Datensatz aus der Datenbank geladen wird, auf Basis der Ergebnisse weitere Datensätze geladen werden, auf Basis dieser Ergebnisse wieder

weitere usw. Im vorliegenden Szenario werden allerdings nur die Kunden abhängig vom Auftrag geladen, es liegt nur eine milde Form des Circuitous Treasure Hunt vor, der wegen seiner häufigen Ausführung aber trotzdem die festgestellten Konsequenzen hat,

c. Geben Sie mindestens drei Entwurfsalternativen an, die die Performanz der problematischsten Stellen verbessern können. Begründen Sie.

   Vergleiche für die Begründung auch Lösung von b.

   1. Dem Benutzer mehrere Positionen gleichzeitig anzeigen (Fast Path).
   2. Bei der Statistikgenerierung alle Kunden in einem Batch laden und dann zuordnen (Batching).
   3. Die Schnittstelle der DBAccess-Komponenten um eine neue Methode erweitern, die die Verträge mit den zugeordneten Kunden zurückliefert und intern diese natürlich performant lädt, mit z.B. Batching (Coupling).
   4. Einen Cache einbauen, der die für die Statistik benötigten Daten bereits enthält, da diese immer dieselben sind.
   5. Nur Aufträge aus der Datenbank laden, auf die die angegebenen Parameter des Benutzers zutreffen (z.B. Zeitraum).

Ausgabe: 16.05.2007

Abgabe der Lösungen: 22.05.2007

Vorstellung der Musterlösung: 25.05.2007

Übungsblatt 4

Ziel der Übung: Weitergehende Performanzanalysen mit SPE-ED.

Allgemeines

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

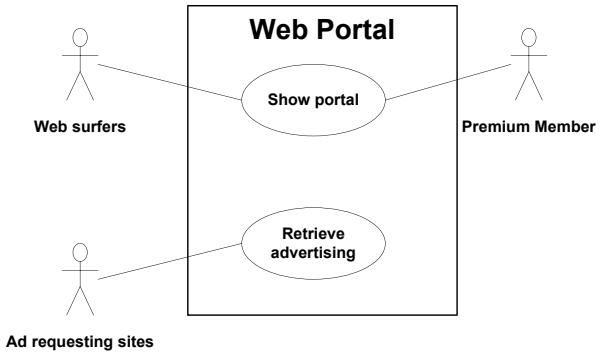
Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Werkzeuge

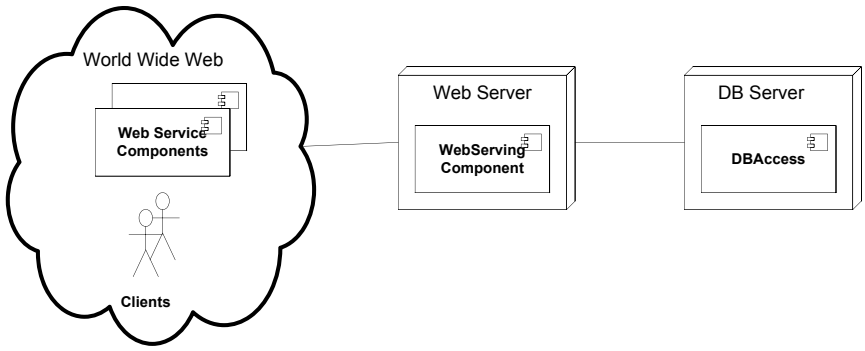
Für die Bearbeitung der Aufgaben benötigen Sie das SPE-ED Werkzeug, das Sie in der zweiten Übung bereits verwendet haben.

Aufgaben

Ein neues Web-Portal mit Nachrichten, Wetter und natürlich Werbung soll entwickelt werden. Um zu prüfen, ob die geplanten Hardware-Ressourcen ausreichen, um eine genügend große Anzahl von Benutzern zu bedienen, soll eine Performanzanalyse durchgeführt werden. Neben dem Aufruf des Portals sollen über den verwendeten Webserver auch Werbungsbanner für andere Websites angeboten werden. Das folgende Diagramm stellt diese beiden Anwendungsfälle, die als die performanzkritischen Szenarien identifiziert worden sind, dar.



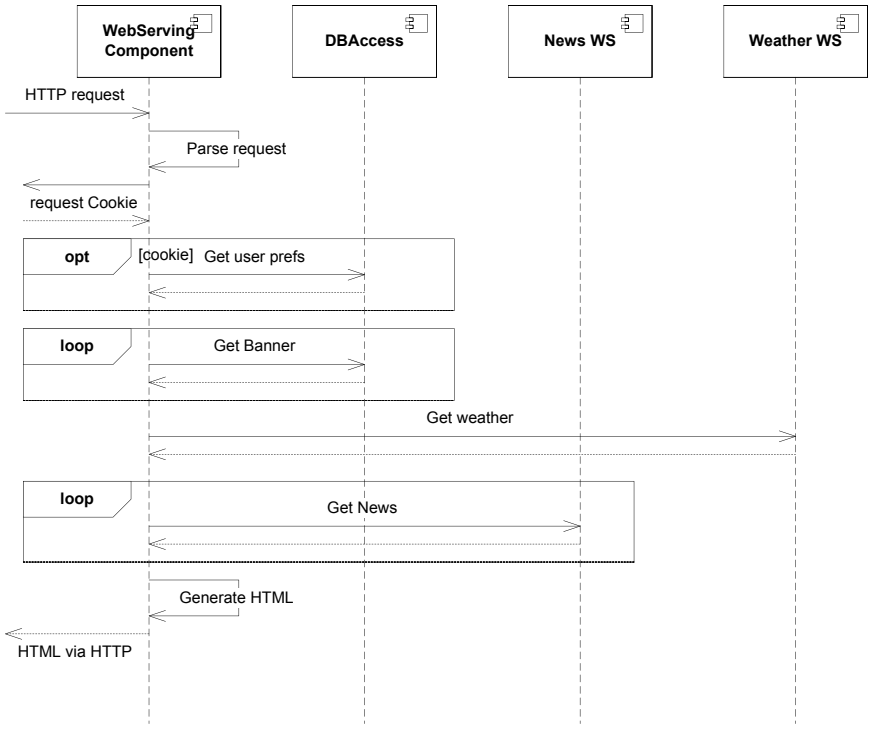
Das Web-Portal ist auf zwei Servern eingesetzt, einer für den benötigten Webserver und einer für die interne Datenbank. Weiterhin werden externe Web Services aufgerufen, die Nachrichten und Wetterdaten liefern. Das folgende Diagramm zeigt das Deployment.



Die einzelnen Web Services für Nachrichten und Wetter sind von anderen Anbietern angeboten, sie stehen zur Entwurfszeit noch nicht fest und werden sich ggf. auch zur Laufzeit ändern. Hier sind keine Daten über die Einsatzumgebungen bekannt, sondern nur Schätzungen, wie lange sie für die Bearbeitung einer Anfrage benötigen.

Die beiden Szenarien laufen wie folgt ab:

Szenario Show Portal:



Die WebServingComponent erhält eine Anfrage über HTTP und parst diese. Der Webbrowser des Benutzers wird nach gespeicherten Cookies für diese Seite gefragt. Liegt ein Cookie vor, so werden die Einstellungen für diesen Benutzer aus der Datenbank geladen, die benötigte Nachricht ist nur ca. 1 KB groß. Im nächsten Schritt werden die Werbebanner geladen. Die Anzahl der Werbebanner hängt von den Einstellungen des Benutzers ab, denn der Benutzer kann seine bevorzugte Auflösung angeben, so dass unterschiedlich viele Inhalte angezeigt werden können.

Danach wird das Wetter bei einem Web Service abgefragt. Die Daten aller Web Services sind unbekannt bis auf die ungefähre Antwortzeit, deswegen wird der Web Service nur durch eine Verzögerung modelliert, und keine Rücksicht auf Auslastung etc. genommen.

In einer Schleife werden die Nachrichten von verschiedenen anderen Web Services geholt (auch hier kann nur die Verzögerung modelliert werden). Die Anzahl der Nachrichten hängt ebenfalls von den Einstellungen des Benutzers ab.

Schließlich wird aus den gesammelten Daten eine Website in HTML generiert und zurück an Benutzer geschickt.

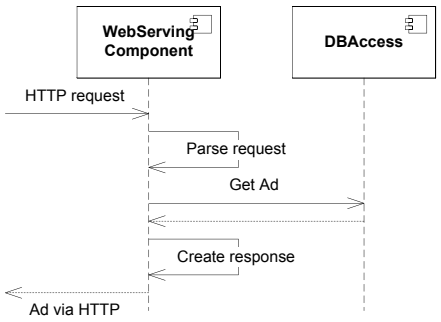
Es wird damit gerechnet, dass ungefähr ein Benutzer pro 5 Sekunden die Seite besucht. Dreiviertel der Benutzer haben bereits ein Profil auf der Seite haben. Von diesen Benutzern hat die Hälfte eine größere Auflösung eingestellt, so dass mehr Anzeigen und Nachrichten angezeigt werden.

Normalerweise werden fünf Anzeigen pro Seite angezeigt, bei einer größeren Auflösung sieben. Eine Anzeige ist im Durchschnitt 20 KB groß. Weiterhin werden normalerweise 8 Nachrichten angezeigt, bei einer größeren Auflösung 15. Tipp: Sie können den gewichteten Mittelwert für die Anzahl der benötigten Schleifendurchläufe verwenden.

Die Abfrage der Nachrichten Web Services benötigt 0,3 Sekunden, die Web Services für das Wetter sind mit 0,4 Sekunden etwas langsamer.

Szenario Werbung abfragen:

Der zweite Anwendungsfall ist das Abfragen von einzelnen Werbebannern (Advertising bzw. „Ad“) durch andere Webserver oder Clients. Er ist im folgenden Diagramm dargestellt.



Dieser Anwendungsfall wird von vielen Benutzern gleichzeitig abgefragt, man rechnet mit durchschnittlich 2 Benutzern pro Sekunde.

Daten zur Hardware-Umgebung

Die CPU des Webservers hat eine Taktrate von 2 GHz. Gehen Sie davon aus, dass eine Instruktion nur im Schnitt 0,8 Zyklen benötigt. Sie können jeweils 10.000 Instruktionen zusammenfassen, indem Sie dafür eine Berechnungseinheit (Work Unit) ansetzen. Es wird

damit gerechnet, dass pro Bearbeitungsschritt maximal 10 Berechnungseinheiten benötigt werden. Sie müssen allerdings abschätzen, wo in diesem Rahmen die Anforderungen eines konkreten Bearbeitungsschritts liegt. Die CPU des Datenbankservers kann ebenso abstrahiert werden, die Taktrate ist allerdings nur 1GHz.

Sie können das mitgelieferte Webserver Facility Template verwenden, müssen hier aber noch die ServiceTimes der einzelnen Devices anpassen und wieder speichern. Leiten Sie den DB Server wiederum davon ab, ändern Sie die Service-Time der CPU und speichern Sie es als DBServer Facility Template. Achten Sie daraus, das DB Server Template nur für das eine Szenario zu übernehmen. Wählen Sie also bei der Frage, ob Sie es für alle Szenarien übernehmen wollen, nicht „Ok“, sondern „Create“ aus.

Das Netzwerk zwischen Webserver und Datenbankserver überträgt 1 GBit/s, die Latenzzeit 1ms. Entscheiden Sie, ob Sie als Software Resource Requirement die Größe der zu übertragenen Daten in KB, oder die Anzahl der Nachrichten oder beides angeben, und begründen Sie Ihre Entscheidung.

Für die Verzögerung durch die Web Services können Sie einen Delay in Sekunden als Software Resource Requirements angeben, und diesen auch 1:1 in Computer Resource Requirements abbilden. Die Anbindung des Web Servers an das Internet ist schnell genug, so dass sich der Durchsatz nicht bei Konkurrenzsituationen ändert, die Anbindung ist immer schnell genug.

Der Datenbankserver benötigt für jeden Zugriff auf die Daten zwei Festplattenzugriffe. Gehen Sie davon aus, dass die Festplatte durchschnittlich 25 MB pro Sekunde lesen kann, aber eine Latenzzeit von 0,8 ms hat. Entscheiden Sie sich auch hier für Software Resource Requirements und begründen Sie. Neben den Festplattenzugriffen werden für einen Datenbankzugriff 30 Berechnungseinheiten (wie oben angegeben) angesetzt.

Die Internetanbindung der Clients ist typischerweise DSL 1000, es werden also 1024 Kbit/s übertragen. Die Latenzzeit beträgt im Mittel 200ms. Ein Cookie hat eine Durchschnittsgröße von 2KB.

Bedenken Sie, dass es sich um ein verteiltes System handelt, und Sie für die Abläufe auf den einzelnen Servern getrennte Szenarien angeben müssen (Caller – Callee, vgl. ergänzten Foliensatz). Daher werden auch Synchronisationsknoten benötigt. Falls es aufgrund der Einschränkungen der Menge der Subgraphen nicht möglich ist, für jede Synchronisation zwischen den Servern Synchronisationsknoten einzuführen, können Sie auch Basisknoten mit den entsprechenden berechneten Delay ausstatten (da die Synchronisationsknoten ohnehin nicht weiter ausgewertet werden, sondern der Illustration dienen).

Da der Aufruf des Datenbankszenarios in einer Schleife geschieht, müssen Sie die Ankunftsrate im Szenario des Datenbankservers entsprechend erhöhen. Beispiel: Bei einer Ankunftsrate am Webserver von 1 Benutzer pro Sekunde und einer dort vorliegenden Schleife mit durchschnittliche 10 Durchläufen, in der jeweils das Datenbankszenario abgefragt wird, müssen Sie die Ankunftsrate für das Datenbankszenario als 10 Benutzer pro Sekunde angeben.

Performanzziel

Die Anzeige des Portals vom Eintreffen des HTTP Requests bis zum Abschicken der generierten Seite soll nicht länger als 3 Sekunden benötigen. Sie müssen also die Zeit, um den HTTP Request zu übertragen, und die Zeit, die für die Übertragung der generierten Seite benötigt wird, nicht berücksichtigen. Die Zeit für die Abfrage des Cookies ist allerdings relevant. Die Abfrage eines Werbebanners soll nicht länger als 0,1 Sekunde benötigen, ebenfalls vom Eintreffen des HTTP Requests bis zum Abschicken des Werbebanners.

Ihre Aufgaben

a. Nennen Sie das Projekt „Uebung4“. Legen Sie für diese Szenarien Software Execution Models an. Sie können logisch zusammenhängende Knoten zusammenfassen, geben Sie ihnen aber aussagekräftige Namen. Machen Sie weiterhin Gebrauch von der Schachtelung des Ausführungsgraphen (Expand). Denken Sie aber daran, dass nur drei Subgraphen in dieser Version des SPE-ED Werkzeugs angelegt werden können. Geben Sie weiterhin die Overhead Matrix an und begründen Sie Ihre Auswahl.

b. Lösen Sie die einzelnen Modelle für einen einzelnen Benutzer, für mehrere Benutzer in einem Szenario und per Simulation aus dem Show Portal Szenario. Betrachten Sie die Antwortzeiten. Geben Sie für die einzelnen Szenarien an, ob das Performanzziel in den einzelnen Stufen erreicht werden kann sowie die ermittelten Werte. Analysieren Sie, wo ggf. Probleme verursacht werden, und schlagen Sie eine Verbesserung der Architektur vor (auch wenn keine Probleme erkennbar sind).

Geben Sie Ihre Lösungen zu a. und b. in einem separaten Dokument namens uebung04 ab, in einem der erlaubten Dateiformate.

Legen Sie das gesamte SPE-ED Verzeichnis ohne die .exe Datei in Ihren SVN Repositories ab. Falls Sie die SPE-ED Installation aus dem vorigen Übungsblatt verwenden wollen, können Sie die Lösung auch in das Verzeichnis zu Übung 2 hochladen. Legen Sie dann aber unbedingt im Verzeichnis „uebung04“ eine Textdatei mit einem entsprechenden Hinweis an.

Tipp: Sie können Subgraphen in anderen Szenarien wiederverwenden, indem Sie einen Knoten erweitern (expand) und, bevor Sie Knoten in den neuen Subgraph einfügen, Edit->Include… und dort den gewünschten Subgraphen auswählen.

1. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:

   Aufgabe 1 SPE-ED:

   a) Zeitangabe

   b) Zeitangabe

Ausgabe: 16.05.2007

Abgabe der Lösungen: 22.05.2007

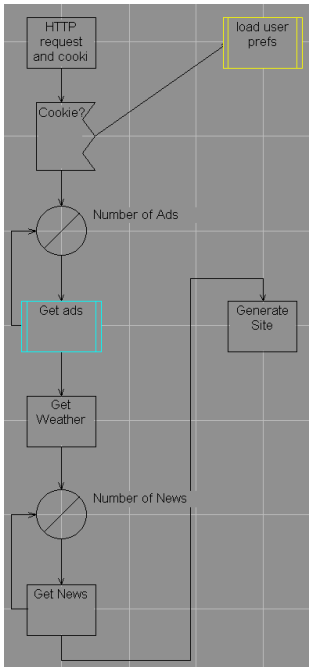
Vorstellung der Musterlösung: 25.05.2007

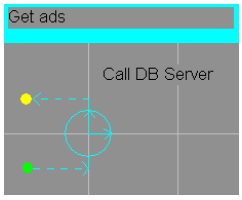
Musterlösung Übungsblatt 4

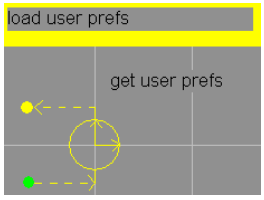
Software Execution Graphs

Es sind drei Szenarien zu erstellen: Show Portal, Retrieve Advertising und DB Process, der von beiden vorherigen verwendet wird.

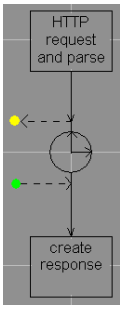
Show Portal:



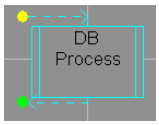


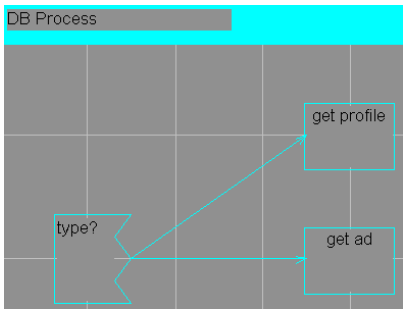


Retrieve Advertising:



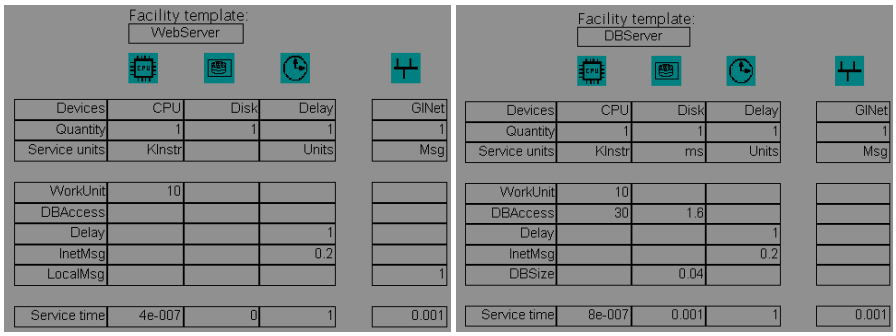
DB Process:





Overhead Matrix

Die Overhead-Matrizen waren selbst zu erstellen und könnten so aussehen:



Facility template: WebServer

Devices	CPU	Disk	Delay	GlNet
Quantity	1	1	1	1
Service units	KInstr		Units	Msg
WorkUnit	10			
DBAccess				
Delay			1	
InetMsg			0.2	
LocalMsg				1
Service time	4e-007	0	1	0.001

Facility template: DBServer

Devices	CPU	Disk	Delay	GlNet
Quantity	1	1	1	1
Service units	KInstr	ms	Units	Msg
WorkUnit	10			
DBAccess	30	1.6		
Delay			1	
InetMsg			0.2	
DBSize		0.04		
Service time	8e-007	0.001	1	0.001

Herleitung der Werte:

CPU: Taktrate Webserver 2GHz. Dabei benötigt eine Instruktion nur 0,8 Zyklen. Es können also 2G / 0,8 = 2,5G Instruktionen pro Sekunde verarbeitet werden. Angabe in der Matrix ist in Kiloinstruktionen. Also noch 2,5 M KInstr / Sek. Für eine Sekunde ist das umgerechnet 1 / 2,5 * 10^6 = 4*10^{-7}. Für den DBServer ist bei halber Taktrate die Service Time entsprechend das Doppelte: 8*10^{-7}.

Disk: Vorgabe: Gehen Sie davon aus, dass die Festplatte durchschnittlich 25 MB pro Sekunde lesen kann, aber eine Latenzzeit von 0,8 ms hat. Es werden maximal im Mittel 20 KB ausgelesen. Bestimmung des Einflusses: Die oben genannte Geschwindigkeit: 1 / (25 * 10^3MB/s) = 0,04 ms pro KB, also benötigt das Auslesen von 20 KB auch 0,8 ms. Die Latenzzeit ist ähnlich, deswegen sind beide Werte also relevant. Es werden daher zwei Software Resource Requirements angegeben, einmal die Anzahl der Datenbankzugriffe und einmal die Größe der zu lesenden Daten. Für den Web Server ist diese Angabe nicht relevant, daher wird die Angabe weggelassen. Die Service Time der Festplatte wird beim DB Server mit 1ms angegeben, und die Software Resource Requirements darauf abgebildet: DBAccess gibt die Menge der Datenbankzugriffe an und beinhaltet auch die nötigen Work Units. Jeder Datenbankzugriff beinhaltet zwei Festplattenzugriffe à 0,8 ms, daher die Abgabe von 1,6 ms. DBSize gibt die Menge der zu lesenden Daten in KB an und wird daher auf 0,04ms abgebildet.

Delay: Die Abbildung wird eins-zu-eins auf Sekunden vorgenommen.

InetMsg: Vorgabe: Die Internetanbindung der Clients ist typischerweise DSL 1000, es werden also 1024 Kbit/s übertragen. Die Latenzzeit beträgt im Mittel 200ms. Ein Cookie hat eine Durchschnittsgröße von 2KB. Bestimmung des Einflusses: 1024 Kbit/s = 128 KByte/s, also wird ein Cookie in 0,015 Sekunden übertragen. Latenzzeit ist 0,2 Sekunden, daher wieder um den Faktor 10 höher. Es kann nur die Latenzzeit betrachtet werden. Daher wieder als SRR die Anzahl der Nachrichten über das Internet, Abbildung auf ein Delay von 0,2 Sekunden, da laut Vorgabe keine nennenswerte Auslastung der Internetverbindung vorliegt.

LocalMsg: Vorgabe: Das Netzwerk zwischen Webserver und Datenbankserver überträgt 1 GBit/s, die Latenzzeit ist 1ms. Es werden hier auch maximal im Mittel 20 KB übertragen, die (rechne, rechne) in 0,16 Millisekunden übertragen können werden. Auch hier ist die Latenzzeit also wieder höher und wird allein verwendet. Die Ressource GlNet (Global Network) wird belastet, die von allen Facilities geteilt wird.

Für die Auslastung der Netzwerkverbindung ist die Latenzzeit allerdings problematisch, da in einem durch Router o.ä. unterteilten Netzwerk auch mehrere Nachrichten wie in einer Pipeline verschickt werden können. Die berechneten Auslastungswerte sind also mit Vorsicht zu interpretieren. Man könnte prüfen, ob die berechnete Auslastung ungefähr mit der durch Kollisionen etc. vergrößerten echten Auslastung übereinstimmt.

Schleifendurchläufe

Die Anzahl der geladenen Werbebanner und Nachrichten variiert je nach Benutzungsprofil. Daher muss der Mittelwert erst einmal berechnet werden.

Vorgaben: Dreiviertel der Benutzer haben bereits ein Profil auf der Seite haben. Von diesen Benutzern hat die Hälfte eine größere Auflösung eingestellt, so dass mehr Anzeigen und Nachrichten angezeigt werden. Normalerweise werden fünf Anzeigen pro Seite angezeigt, bei einer größeren Auflösung sieben. Eine Anzeige ist im Durchschnitt 20 KB groß. Weiterhin werden normalerweise 8 Nachrichten angezeigt, bei einer größeren Auflösung 15.

Also für die Anzahl der Schleifendurchläufe für die Banner:

$7 * 0{,}75 * 0{,}5 + 5 * (1 – 0{,}75 * 0.5) = 5{,}75$

Für die Nachrichten:

$15 * 0{,}75 * 0{,}5 + 8 * (1 – 0{,}75 * 0.5) = 10{,}625$

Ankunftsraten

Ankunftsrate Show Portal ist 0,2 jobs/s. Ankunftsrate Retrieve Advertising ist 2 jobs/s. Pro Aufruf von Show Portal wird die Datenbank im Mittel 5,75-mal für ein Werbebanner und 0,75-mal für das Profil des Benutzers angefragt, insgesamt also 6,5-mal. Die Ankunftsrate für DB Process ist somit 6,5 * 0,2 jobs/s + 2 jobs/s = 3,3 jobs/s.

Bedingungen

Im DB Process wird entschieden, ob ein Benutzerprofil oder ein Werbebanner geladen wird. Insgesamt kann dies allerdings auch anders modelliert werden, indem es in zwei Szenarien aufgeteilt wird. Die Wahrscheinlichkeiten, ob ein Banner oder ein Profil geladen wird, hängt von den Ankunftsraten der anfragenden Prozesse und der Häufigkeit des Aufrufs innerhalb der Prozesse zusammen.

Im Szenario Show Portal gilt erst einmal:

$p_{sp}(\text{getProfile}) = 1 / (5{,}75 + 1) = 0{,}149$ und $p_{sp}(\text{getAd}) = 1 - p_{sp}(\text{getProfile}) = 0{,}851$

Im Szenario Retrieve Ads wird nur getAd nachgefragt, mit einer Ankunftsrate von 2 jobs/s im Gegensatz zu 0,2 jobs/s von Show Portal. Insgesamt gilt also:

$p(\text{getProfile}) = p_{sp}(\text{getProfile}) * 0{,}2 / 2{,}2 = 0{,}014$ und $p(\text{getAd}) = 1 - p(\text{getProfile}) = 0{,}986$

Delays für Synchronisation

Mit den bisherigen Angaben kann ein Delay von 0,002431 s für den DB Process ermittelt werden (Contention Solution, aber nicht Simulation) und als Delay in den anderen Software Execution Graphs eingesetzt werden. Es muss nicht simuliert werden, weil der DB Process mit keinem anderen Szenario die Ressourcen des DBServers teilen muss.

Ergebnisse

Show Portal außerhalb der Vorgaben mit 4,0184 Sekunden Antwortzeit. In der Simulation ist die Antwortzeit sogar noch höher.

Retrieve Advertising: Antwortzeit 0,004774 Sekunden und DB Process: Antwortzeit 0,002431 Sekunden. Bei der Simulation starke Schwankungen bei 20.000 Durchläufen, teilweise kein Erreichen der Konfidenz.

Ausgabe: 23.05.2007

Abgabe der Lösungen: 29.05.2007

Vorstellung der Musterlösung: 01.06.2007

Übungsblatt 5

Ziel der Übung: Einüben des Repository-Konzeptes von Palladio (Aufgabe 1), weiterführende Aufgabe SPE-ED (Aufgabe 2)

Allgemeines

Die Übungsgruppeneinteilung wird nur noch alle zwei Wochen neu erstellt, Sie bearbeiten diesen Zettel also mit Ihrem Übungspartner vom letzten Mal.

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Werkzeuge

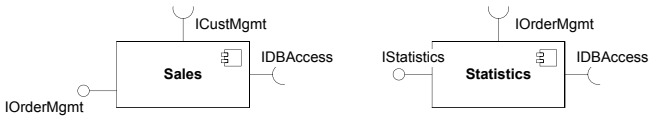
Für die Bearbeitung der Aufgaben benötigen Sie das SPE-ED Werkzeug, das Sie in der zweiten Übung bereits verwendet haben. Weiterhin wird die aktuelle Version der PCM Bench benötigt, die ab morgen im Wiki verlinkt sein wird. Sie können also Aufgabe 1 erst ab Donnerstag, nachdem die neue Version verfügbar ist, bearbeiten. Sie müssen diese dann erneut herunterladen und auch die PCM Plugins neu hinzufügen, wie auf Übungszettel 2 angegeben. Weiterhin benötigen Sie das im Wiki verlinkte Repository mit vorbereiteten primitiven Datentypen (PrimitiveTypes.repository).

Aufgaben

1. PCM: Sie haben das Informationssystem CBIS bereits in Übung 3 bearbeitet. Nun sollen die dort vorgestellten Konzepte auch im PCM modelliert werden. In der Zwischenzeit haben sich die Architektur und der Entwurf der einzelnen Komponenten aufgrund der festgestellten Performanzmängel verändert. So wurde beispielsweise eine Komponente eingeführt, die einen lokalen Cache für die generierten Statistiken ausliest. Weiterhin wurden die Komponenten Sales und Statistics überarbeitet und liegen nun in als FastSales und FastStatistics in einer weiteren Version vor: Die Abläufe wurden verschlankt, die Benutzer können bei dem Anlegen eines Auftrags gleich mehrere Positionen angeben. Bei der Aktualisierung des Caches oder der Abfrage noch nicht vorgehaltener Statistiken werden nicht alle Kunden und Aufträge einzeln geladen, sondern in Batches.

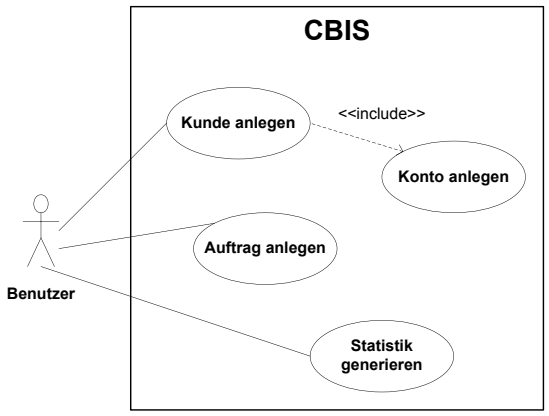
Die alten Versionen der Komponenten bleiben weiterhin zu Dokumentationszwecken erhalten:

Weitere Komponenten im Repository:



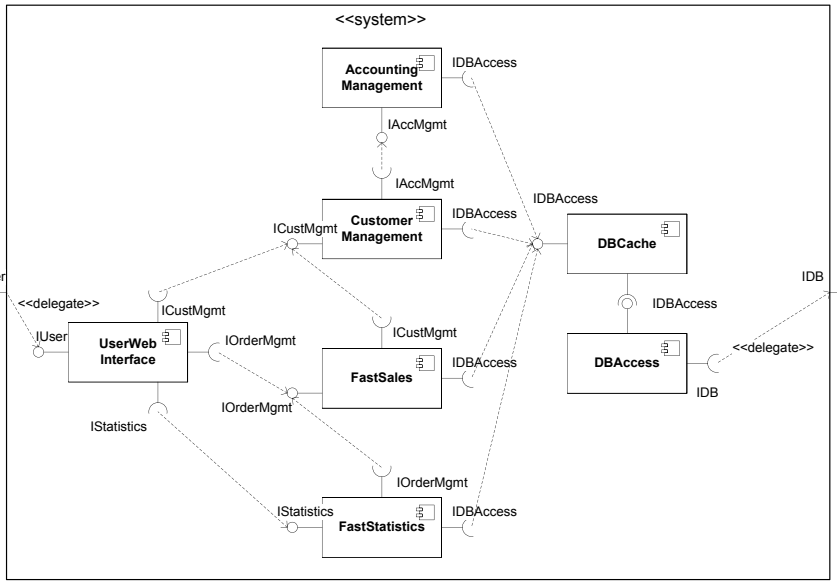
Die Anwendungsfälle bleiben weiterhin dieselben:

Anwendungsfälle:



Entwurf der Architektur:

Die einzelnen Funktionsbereiche werden weiterhin in einzelnen Komponenten realisiert: CustomerManagement, AccountingManagement, FastSales und FastStatistics. Zusätzlich ist die Komponente DBAccess für den Datenbankzugriff zuständig, sie abstrahiert von den SQL Anfragen an die Datenbank und stellt dem restlichen System eine objektorientierte Schnittstelle bereit. Die Komponente UserWebInterface wandelt die Anfragen, die über HTTP eingehen, in Anfragen an die einzelnen Funktionskomponenten um. Zusätzlich wurde eine Komponente DBCache vor DBAccess geschaltet, die einige Anfragen an die Datenbank zwischenspeichert und so teure Datenbankzugriffe erspart.



Für einige Schnittstellen sind bereits weitere Informationen zu den Signaturen und den darin verwendeten Datentypen verfügbar.

Die Schnittstelle ICustMgmt verfügt u.a. über die Methoden

- String createCustomer(List<String> customerData), die die Kundennummer zurückgibt
- Customer getCustomer(String customerNo)
- List<Customer> getAllCustomers()

Die Schnittstelle IDBAccess verfügt u.a. über die folgenden Methoden, dabei ist Storable ein Interface, dass von allen in der Datenbank zu speichernden Datentypen implementiert wird. Sie können es als Composite Data Type ohne Inner Declarations modellieren.

- String create(Storable data), die eine Identifikation (z.B. die Kundennummer) zurückgibt
- List<Storable> getAll(String objecttype)
- Storable get(String id, String objecttype), die das geladene Objekt zurückgibt

Die Schnittstelle IAccMgmt verfügt u.a. über die Methoden

- String createCustomerAccount(String customerNo), die die Kontonummer zurückgibt.

Dabei besteht der zusammengesetzte Datentyp Customer aus den folgenden Attributen:

- String customerNo
- String firstname
- String lastname
- String address

Der zusammengesetzte Datentyp Account besteht aus den folgenden Attributen

- String accountNo
- String customerNo
- String accountName
- String accountType

Ihre Aufgaben

a. Modellieren Sie die im CBIS verwendeten Komponenten sowie die alten Versionen in einem PCM Repository. Erstellen Sie für die Komponenten, die sinnvolle gemeinsame Provided und/oder Complete Types haben können, auch diese. Für die anderen brauchen Sie nur die Basic Component Types zu erstellen. Geben Sie die Signaturen wie oben angegeben an. Sie müssen dazu aus den gegebenen Datentypen des PrimitiveTypes.repository neue Datentypen erstellen. Lassen Sie schließlich Ihre Diagramme vom Werkzeug validieren (Diagram -> Validate) um sicherzustellen, dass keine groben Modellierungsfehler enthalten sind.

2. SPE-ED: Für das Web Portal System aus dem vorigen Übungsblatt stellt man sich die Frage, wie viele Benutzer einzelne Werbebanner abfragen können, ohne die Funktionalität des Portals selbst stark zu beeinträchtigen (Szenario Retrieve Advertising / Werbung abfragen).

   Sie sollen diese Untersuchung mit SPE-ED durchführen. Korrigieren Sie dazu zunächst Ihre Lösung zu Übung 4 anhand der Ihnen vorliegenden Musterlösung. Sie müssen nicht exakt dieselben Werte erhalten, insbesondere bei der Simulation ist dies auch gar nicht möglich. Sie sollten jedoch die Berechnungen der Wahrscheinlichkeiten für Verzweigungen und der Anzahl der Schleifendurchläufe sowie die Overhead-Matrix überprüfen und ggf. anpassen. Weiterhin sollten Sie alles korrigieren, was zu Punktabzug geführt hatte und nicht anders markiert ist.

   a. Finden Sie heraus, wie viele Benutzer in der Sekunde Werbebanner abfragen können, ohne die Antwortzeit des Anwendungsfalls Show Portal um mehr als 10% zu verändern. Bedenken Sie auch die dadurch höhere Ankunftsrate am DB Prozess. Geben Sie in Ihrer Abgabe an, welche Untersuchungen Sie durchgeführt haben, um den Wert festzustellen.

   b. Ab einem gewissen Zeitpunkt kann eine Ressource die eingehenden Anfragen nicht mehr schnell genug abarbeiten, und die Warteschlange wird immer länger. Finden Sie heraus, ab welcher Benutzerzahl dies in diesem System geschieht und geben Sie ebenfalls an, wie Sie zu dieser Lösung gekommen sind.

3. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:

   1 a) Zeitangabe

   2)

   a) Zeitangabe

   b) Zeitangabe

Abgabe

Geben Sie Ihre Lösungen zu 1. und 2. in einem separaten Dokument namens uebung05 ab, in einem der erlaubten Dateiformate.

Legen Sie das gesamte SPE-ED Verzeichnis ohne die .exe Datei in Ihren SVN Repositories ab. Falls Sie die SPE-ED Installation aus dem vorigen Übungsblatt verwenden wollen, können Sie die Lösung auch in das Verzeichnis zu Übung 2 hochladen. Legen Sie dann aber unbedingt im Verzeichnis „uebung05“ eine Textdatei mit einem entsprechenden Hinweis an.

Prof. Dr. Reussner, Software-Entwurf und -Qualität, Universität Karlsruhe (TH)
Praktikum Ingenieurmäßiger Software-Entwurf, SoSe 2007

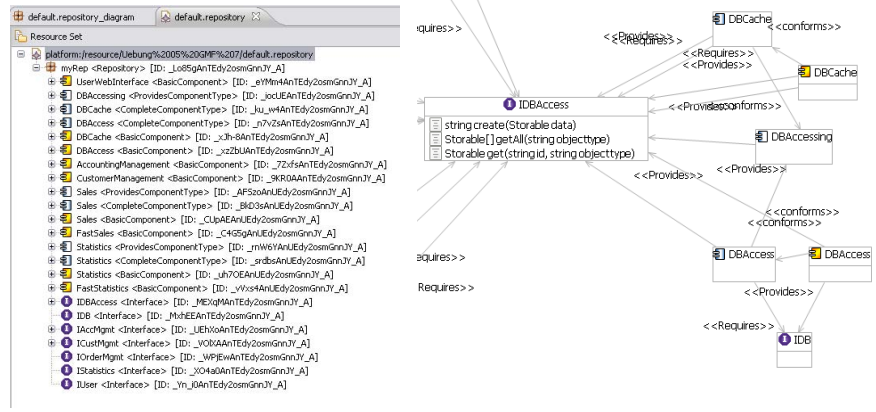
Ausgabe: 23.05.2007

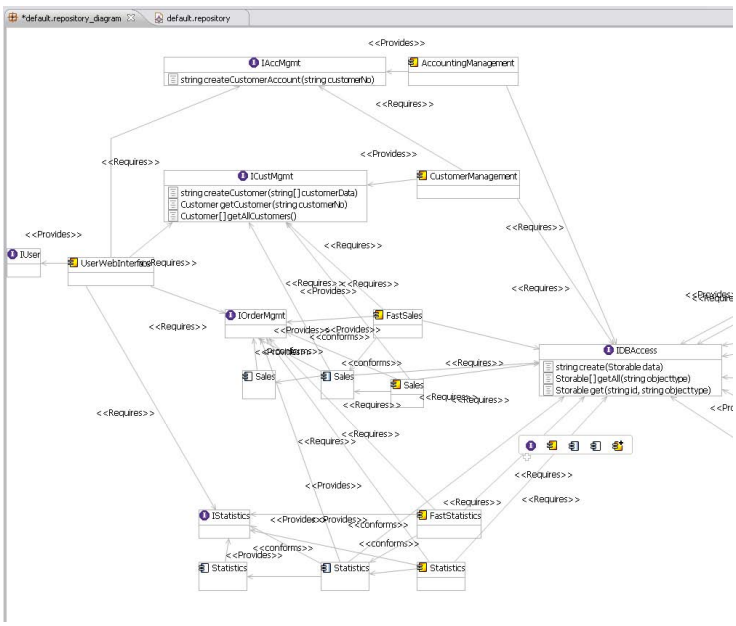
Abgabe der Lösungen: 29.05.2007

Vorstellung der Musterlösung: 01.06.2007

Musterlösung Übungsblatt 5

1.





2.

a) Es sollte die maximale Benutzerzahl für den Anwendungsfall Retrieve Advertising herauszufinden, bei der der Anwendungsfall Show Portal maximal um 10% langsamer wird. Da die Simulationslösung für diese Szenarien ohnehin stark streut, lässt sie sich nicht für diese Untersuchung verwenden. Es müssen die Szenarien also einzeln betrachtet werden.

Zunächst kann durch Erhöhen der Zugriffszahlen für das Szenario herausgefunden werden, dass im DBProcess Szenario die kritischen Ressourcen liegen, und dass sich die höhere Zugriffszahl zwar die Anzahl der benötigten Work Units erhöht, dies aber die Auslastung der CPU kaum beeinträchtigt und sich damit kaum auf den Web Server auswirkt. Kritisch ist die Auslastung des DB Servers.

Nun kann in einem zweiten Schritt untersuch werden, wie lange der DBProcess Aufruf im Show Portal Szenario dauern darf, ohne dass die Antwortzeit des gesamten Szenarios um mehr als 10% steigt. Die Benutzerzahl wird dabei für das Szenario Show Portal nicht angepasst, nur der Delay für die Synchronisationsknoten.

Die bisherige Antwortzeit lag bei 4,0184 s, eine Steigerung von 10 % läge bei 4,42024 s. Mit einem Delay für die Synchronisation mit dem Datenbankserver von 0,05 s erhält man eine Gesamtantwortzeit von 4,3276 s.

Es wird nun untersucht, wie viele Benutzer im DB Process bedient werden können, ohne dass die Antwortzeit sehr viel höher als 0,05 s wird. Dabei muss bedacht werden, dass der Show Portal Prozess 1,3 Benutzer pro Sekunde beisteuert. Weiterhin verändern sich die Wahrscheinlichkeiten, welche Aktion im DBProcess ausgeführt werden sollen: Je höher die Ankunftsrate von Retrieve Advertising, desto größer die Wahrscheinlichkeit von getAd (im Gegensatz zu getProfile).

Zunächst wird der grobe Bereich, in dem die Lösung liegt, bestimmt. Es zeigt sich, dass bei 402,3 Benutzern pro Sekunde ohne Anpassung der Wahrscheinlichkeiten im DB Process die Antwortzeit bei 0,06053 s liegt. Zur Überprüfung wird dieser Wert nochmals in Show Portal eingesetzt, es ergibt sich dort eine Antwortzeit von 4,3961 s, dieser Wert liegt bereits sehr nah an der 10%-Grenze.

Die Wahrscheinlichkeiten für die einzelnen Aktionen (Herleitung vgl. letzte Musterlösung):

Ankunftsrate	p(getProfile)	p(getAd)
350	8.50942E-05	0.99991491
400	7.44628E-05	0.99992554
450	6.61928E-05	0.99993381

Man kann in diesem Bereich also gerundet die Wahrscheinlichkeit p(getProfile) = 0,0001 und p(getAd) = 0,9999 verwenden.

Für die oben durchgeführte Analyse mit 402,3 Benutzern pro Sekunde ergibt sich nun eine Antwortzeit von 0,06957 s im DBProcess Szenario. Eingesetzt in Show Portal ergibt sich dann eine Antwortzeit von 4,4548 s, also mehr als 10 %.

Weitere Untersuchungen für die Feststellung des genauen Werts werden in folgender Tabelle dargestellt:

Ankunftsrate DBProcess	Antwortzeit	Antwortzeit Show Portal
402,3	0,06957 s	4,4548 s
401,3	0,06504 s	4,4256 s
400,3	0,06107 s	4,3996 s

Die Ankunftsrate des Retrieve Advertising Szenarios kann also maximal 399 Benutzer pro Sekunde betragen, ohne dass sich die Antwortzeit von Show Portal um mehr als 10 % erhöht.

b) Auch hier wird wieder der DB Process betrachtet. Bei einer weiteren Erhöhung der Ankunftsrate steigt die Antwortzeit des Szenarios ab einem gewissen Punkt deutlich und sprunghaft an.

Die Ankunftsrate ist für DBProcess angegeben:

Ankunftsrate	Antwortzeit	Auslastung Disk
415,3	0,7247 s	0,7247 s
416,3	2,6325 s	2,6325 s
417,3	239,67 s	239,67 s

Es wird für eine Ankunftsrate von 416,3 Benutzern pro Sekunde eine Antwortzeit von 2,6325 Sekunden angegeben, die durchschnittliche Auslastung der Festplatte liegt hier ebenfalls bei 2,6325, also wächst hier die Warteschlange kontinuierlich an. Das Ergebnis zur Antwortzeit muss hinterfragt werden: Wenn die Warteschlange kontinuierlich wächst, steigt auch die Antwortzeit mit der Zeit, spätere Anfragen werden immer höhere Anfragen haben, und bei einer theoretisch unbeschränkten Länge der Warteschlange strebt auch die mittlere Antwortzeit gegen unendlich. Womöglich kommt das Ergebnis von SPE-ED dadurch zustande, dass die Annahme gemacht wird, dass der Durchsatz des Gesamtsystems immer gleich der Ankunftsrate ist (so dass die in der Vorlesung vorgestellten Berechnungen durchgeführt werden können). Diese Annahme ist in unserem Fall jedoch nicht mehr zulässig. Trotzdem bleibt die Erkenntnis, dass 416,3 Benutzer pro Sekunde nicht mehr abgearbeitet werden können.

Ab 417,3 Benutzern pro Sekunde ist die Auslastung der Festplatte stark gewachsen und wird mit 239,67 s angegeben. Auch weitere Erhöhungen ändern nichts daran, dieser Wert repräsentiert also das Maximum bei SPE-ED und weist auf eine immer weiter wachsende Warteschlange hin.

Es können also maximal 414 Benutzer Werbung abfragen, bevor die Auslastung der Festplatte zu sehr steigt.

Ausgabe: 30.05.2007

Abgabe der Lösungen: 05.06.2007

Vorstellung der Musterlösung: 08.06.2007

Übungsblatt 6

Ziel der Übung: Erstellen von Service Effekt Spezifikationen im PCM

Allgemeines

Zu dieser Übung gibt es eine neue Übungsgruppeneinteilung.

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Werkzeuge

Für die Bearbeitung der Aufgaben benötigen Sie die PCM Bench und die im Wiki verfügbare Musterlösung zu Übung 5, die auch bereits die weiteren Informationen für einen Analyse enthält. Die neue Version der Plugins, die zur Bearbeitung nötig ist, wird erst am Abend verfügbar sein und im Wiki angekündigt werden.

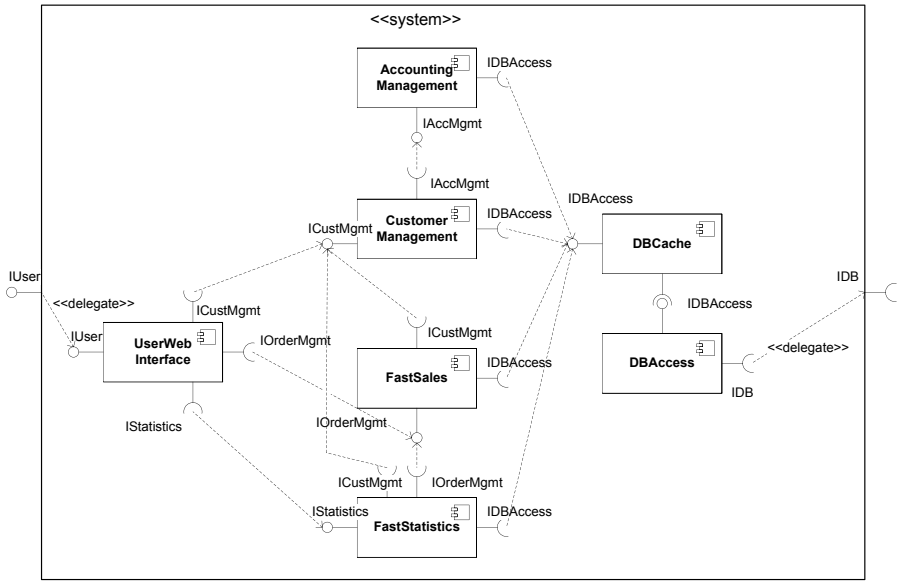
Denken Sie daran, bei dem Arbeiten mit der PCM Bench keine zwei Diagramme gleichzeitig geöffnet zu halten und zwischendurch öfter zu speichern.

Aufgaben

1. PCM: Sie haben das Informationssystem CBIS bereits in den Übungen 3 und 5 bearbeitet. Nun sollen auch die Abläufe selbst im PCM modelliert werden.

   Zur Übung 5 haben sich die Architektur und der Entwurf der einzelnen Komponenten aufgrund der festgestellten Performanzmängel verändert. So wurde beispielsweise eine Komponente eingeführt, die einen lokalen Cache für die generierten Statistiken ausliest. Weiterhin wurden die Komponenten Sales und Statistics überarbeitet und liegen nun in als FastSales und FastStatistics in einer weiteren Version vor: Die Abläufe wurden verschlankt, die Benutzer können bei dem Anlegen eines Auftrags gleich mehrere Positionen angeben. Bei der Aktualisierung des Caches oder der Abfrage noch nicht vorgehaltener Statistiken werden nicht alle Kunden und Aufträge einzeln geladen, sondern in Batches. Die Architektur und die Anwendungsfälle sind auf Übungsblatt 5 angegeben und bleiben bis auf eine kleine Änderung unverändert.

Architektur



Die Komponente FastStatistics enthält nun zusätzlich die Schnittstelle ICustMgmt. Der Complete und Provides Typ wurde nicht angepasst.

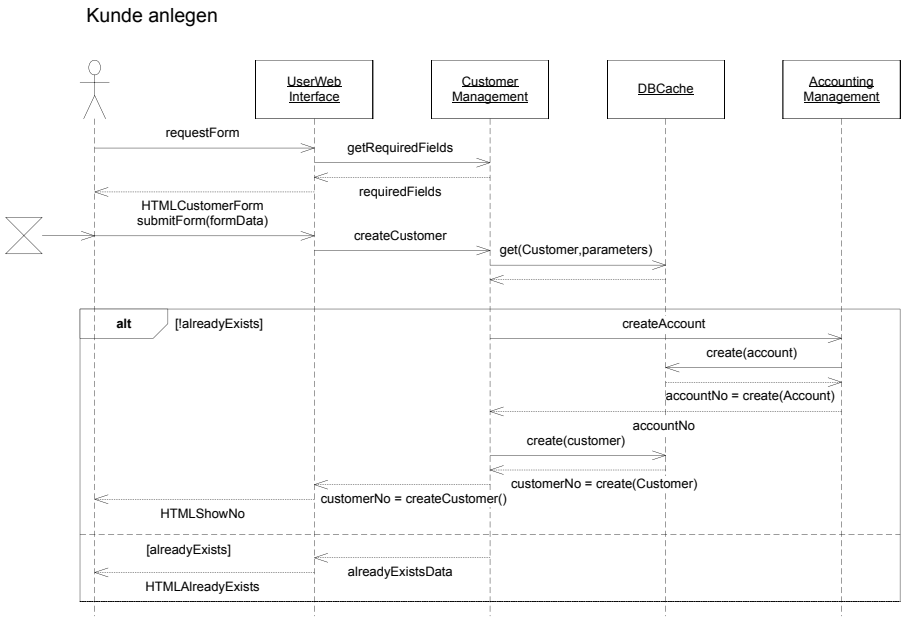
Entwurf der Komponenten:

Die Komponenten wurden bereits in Übung 5 im PCM Repository modelliert und liegen nun in einer überarbeiteten Form im Wiki vor. Für die einzelnen Schnittstellen müssen nun noch die internen Abläufe als Service Effekt Spezifikationen angegeben werden. Die Abläufe haben sich aufgrund der Überarbeitungen verändert und wurden weiterhin verfeinert. Sie sind im Folgenden als Sequenzdiagramme angegeben.

Dabei ist das Verhalten des DBCaches beim Laden nicht mit modelliert, um die Diagramme nicht zu verkomplizieren. Beim Speichern ist das Verhalten beispielhaft im Sequenzdiagramm Auftrag anlegen modelliert, im Sequenzdiagramm Kunde anlegen wurde dies auch zur Vereinfachung weggelassen. In den Service Effekt Spezifikationen soll jedoch auch folgendes Verhalten mit modelliert werden:

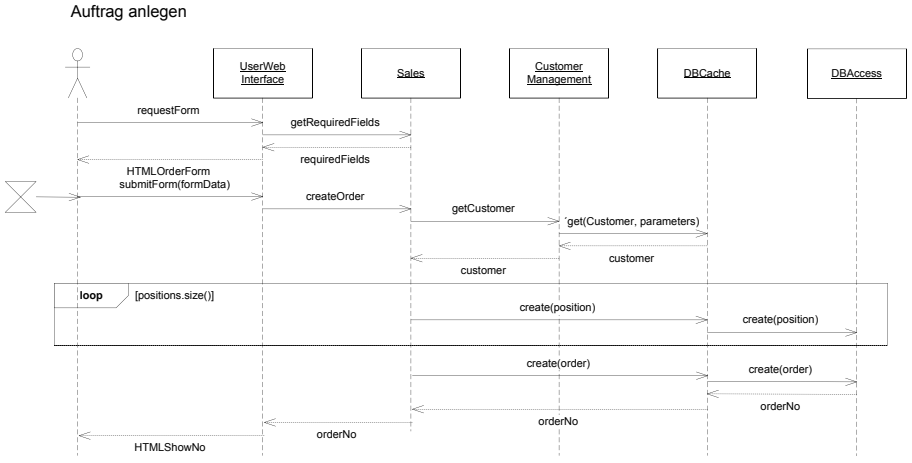
- Bei jedem Speichern von Datensätzen leitet die DBCache Komponente die Anfragen an die DBAccess Komponente weiter
- Bei jedem Laden von Datensätzen besteht eine 50%-ige Wahrscheinlichkeit, dass die angeforderten Daten im DBCache vorliegen. Falls die Daten nicht vorliegen, wird die Anfrage an DBAccess weitergeleitet.

Kunde anlegen:



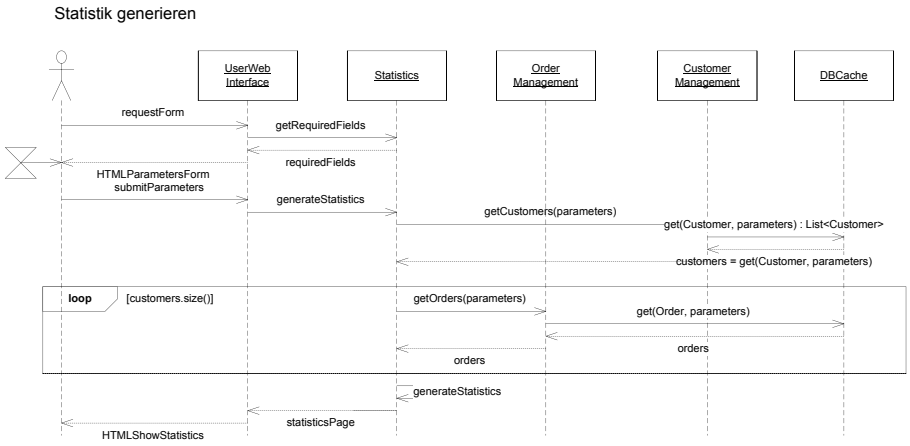
Dieses Szenario hat sich zu vorher kaum verändert, nur dass nun, um zu prüfen, dass ein Kunde noch nicht vorliegt, die get(objecttype, parameters) Methode des DBCaches aufgerufen wird. Wenn diese keine Ergebnisse liefert, existiert der Kunde noch nicht.

Auftrag anlegen:



Hier kann der Benutzer in einem Formular sowohl die Kundennummer als auch verschiedene Positionen angeben, die alle zusammen als `formData` an das UserWebInterface geschickt werden, und der Sales Komponente übergeben werden. Weiterhin wird jede Position einzeln angelegt, weil das Datenbankschema es so erfordert.

Statistik generieren:



Hier werden nun zuerst die für die Statistik benötigten Kunden geladen. Die Parameter, die der Kunde zur Generierung der Statistik angegeben hatte, beispielsweise einen Zeitraum, oder eine regionale Einschränkung, werden berücksichtigt und der DBAccess Komponente als Einschränkungen übergeben. Es werden also nur Kunden geladen, die zu den Parametern passen. Weiterhin werden für jeden Kunden die entsprechenden Aufträge geladen, auch mit Berücksichtigung der Parameter. Dabei wird die Kundennummer als einer der Parameter übergeben.

Bedenken Sie, dass im PCM die Service Effekt Automaten für einzelne Methoden aus den Schnittstellen, nicht für einzelne Anwendungsfälle erstellt werden. Sie müssen daher beispielsweise für den Aufruf requestForm der UserWebInterface Komponente die Fallunterscheidung, welche weitere Komponente aufgerufen werden soll, modellieren. Auch die Methode get(..) der DBAccess Komponente ist betroffen, s. unten. Überlegen Sie also, welche Methoden in mehreren Anwendungsfällen vorkommen und wo die zugehörigen Service Effekt Automaten eine Verallgemeinerung der unterschiedlichen Sequenzdiagramme darstellen müssen.

Verwenden Sie für die Modellierung der Fallunterscheidungen Probabilistic Branches mit den folgenden Wahrscheinlichkeiten:

- Der Anwendungsfall „Kunde anlegen“ wird in 20% der Fälle ausgeführt, der Anwendungsfall „Auftrag anlegen“ in 70% der Fälle und nur in 10% der Fälle wird eine Statistik generiert.
- Eine Anfrage kann zu 50% direkt aus dem Cache beantwortet werden.
- Weiterhin liegen nur 5% der Kunden bereits vor, wenn sie neu angelegt werden sollen.

Beachten Sie, dass die Wahrscheinlichkeit der Anwendungsfälle auch bestimmt, wie wahrscheinlich die einzelnen Formen der Datenbankzugriffe sind: Wann werden bei einem get(..) nur wenige Daten abgerufen, wann viele? Modellieren Sie diese Auswahl im SEFF für DBAccess.get(..) mit Probabilistic Branches.

Für die Modellierung der SEFFs wurden zu den in Übung 5 angegebenen Signaturen weiterhin einige Signaturen ergänzt und einige Signaturen verändert. Beides ist im vorliegenden Modell bereits ein gepflegt:

Neue Signaturen:

Schnittstelle IUser:

- String requestForm(String whichForm)
- String submitForm(List<String> formData)

Schnittstelle IOrderMgmt

- String createOrder(List<String> orderData)
- List<Order> getOrders(List<String> parameters)
- List<String> getRequiredFields(String operation)

Schnittstelle IStatistics

- String generateStatistics(List<String> parameters)
- List<String> getRequiredFields(String operation)

Die Schnittstellen ICustMgmt benötigt ebenfalls die Methode

- List<String> getRequiredFields(String operation)

Geänderte Signaturen:

Die Methode get(..) der Schnittstelle IDBAccess hat sich im Vergleich zum letzten Übungszettel geändert:

- List<Storable> get(String objecttype, List<String> parameters), die eine Liste von passenden Objekten zurückgibt

Die Methode getCustomer der Schnittstelle ICustMgmt wird wie folgt angepasst:

- List<Customer> getCustomer**s**(List<String> parameters)

Die Methoden getAllCustomers der Schnittstelle ICustMgmt und getAll() der Schnittstelle IDBAccess werden nicht mehr benötigt und können gelöscht werden.

Weitere Datentypen:

Diese Datentypen sind neu und auch bereits im Modell angelegt. Der Datentyp Order beinhaltet nur Kopfdaten eines Auftrags und zwar die folgenden:

- String orderNo
- String customerNo
- String orderDate
- String requestedDeliveryDate

Der Datentyp Position enthält folgende Daten:

- String orderNo
- String positionsNo
- String articleNo
- double amount

Resource Demands

Die Resource Demands müssen größtenteils wie in Übung 3 angegeben werden. Dabei ist die Geschwindigkeit der CPU für 1 Work Unit, d.h. 10.000 Kiloinstruktionen, spezifiziert, Sie müssen die Resource Demands an die CPU also in Work Units à 10.000 Kiloinstruktionen angeben. Da die Abstraktionsebene der Software Resource Requirements fehlt, müssen Sie die Anforderung DBAccess auf die Ressourcen CPU und HDD (Festplatte) aufteilen: Ein DBAccess entspricht weiterhin 5 Work Units und 3 Festplattenzugriffen, die direkt als Anforderung an die Ressource HDD modelliert werden können.

Zur Berechnung der Resource Demands benötigen Sie weiterhin die neuen Angaben zur Statistikberechnung. Es werden aufgrund der Berücksichtigung der angegebenen Parameter nun nur noch durchschnittlich 1000 Kunden für die Berechnung der Statistik aus der Datenbank geladen. Weiterhin hat jeder dieser Kunden im Schnitt 2 Aufträge, die zu den Parametern passen.

Wie bei Übung 3 benötigt das Berechnen der Statistik 3 Work Units pro Bestellung, die berücksichtigt wird. Diese Berechnung wird erst durchgeführt, nachdem alle Bestellungen geladen wurden (vgl. Sequenzdiagramm).

Um zu prüfen, ob ein Datenbankeintrag bereits im Cache vorliegt, benötigt die Komponente DBCache 5 Work Units.

Ihre Aufgaben

1. Modellieren Sie die Service Effekt Spezifikationen für das CBIS auf Basis der Musterlösung. Für die alten Versionen der Sales und Statistics Komponente müssen keine SEFFs modelliert werden. Benennen Sie die SEFF Diagramm Dateien nach dem folgenden Schema: „Komponentenname.Methodenname“, dabei können Sie die Namen auch abkürzen, solange sie eindeutig bleiben.

   Denken Sie daran, vor jedem Wechsel zwischen Diagrammen den Stand des aktuellen Diagramms zu speichern, um keine Inkonsistenzen zwischen den verschiedenen Diagrammen zu erhalten.

2. Analysieren Sie das Modell mit der Simulation für einen Benutzer, in dem Sie Run->Open Run Dialog auswählen und hier eine neue Run Configuration für die SimuBench angeben.

   Für die Simulation müssen Sie auswählen, welche Repositories, Systems, etc. Sie für die Simulation verwenden möchten. Wählen Sie das default.repository als Repository File aus. Da in unserem Beispiel jede weitere Art von Modell nur einmal vorkommt, wählen Sie jeweils diese eine Datei aus, die Sie an der Dateiendung erkennen können. Das Usage File hat die Endung .usagemodel. Sie können weiterhin einen Namen für diese Analyse vergeben, im Reiter SimuCom.

   Führen Sie die Simulation durch und betrachten Sie die Ergebnisse in der SimuBench Perspektive. Im Experiments View finden Sie Ihren Run der Simulation mit den verschiedenen Sensordaten. Der Sensor „Response Time of usageScenario“ enthält die Messdaten zum Gesamtsystem. Ziehen Sie den Sensor per Drag and Drop auf die Fläche in der Mitte. Wählen Sie JFreeChart Histogram aus und klicken Sie danach im Property Sheet auf „Update Chart“. Falls das Property Sheet nicht angezeigt wird, klicken Sie zunächst auf das leere Histogramm.

   Fügen Sie einen Screenshot des Histogramms einem Dokument hinzu und geben Sie in 3 – 5 Sätzen an, was aus dem Histogramm geschlossen werden kann. Die Antwort sollte

nicht zu knapp sein, sondern widerspiegeln, dass Sie Zusammenhänge erkennen. Sie braucht aber auch nicht übermäßig ausführlich sein und das Ergebnis bis ins Detail betrachten.

Hinweis: Wenn Sie mehrere Durchläufe machen, müssen Sie die Ansicht in der SimuBench aktualisieren, um auch die neusten Ergebnisse angezeigt zu bekommen.

3. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:
    1. Zeitangabe
    2. Zeitangabe

Abgabe

Das PCM Projekt, ein Dokument mit dem Screenshot der Simulation (im Word, PDF oder OpenOffice Format) sowie die Angaben des Zeitaufwands müssen eingecheckt werden.

Musterlösung Übungsblatt 6

Vgl. auch eingecheckte Musterlösung

Wahrscheinlichkeiten DBAccess.get:

1000 mal Abfrage von ein bis zwei Aufträgen im Szenario generateStatistics (0.1)

1 mal Abfrage von einem Kunden im Szenario createCustomer (0.2)

1 mal Abfrage von einem Kunden im Szenario createOrder (0.7)

1 mal Abfrage von 1000 Kunden im Szenario generate Statistics (0.1)

1000 * 0.1 = 100

1 * 0.2 + 1 * 0.7 = 0.9

1 * 0.1 = 0.1

Also Verhältnis 100 : 0.9 : 0.1, entspricht ungefähr 99 : 0.9 : 0.1 → Wahrscheinlichkeiten 0.99 : 0.009 : 0.001

Ausgabe: 06.06.2007

Abgabe der Lösungen: 12.06.2007

Vorstellung der Musterlösung: 22.06.2007

Übungsblatt 7 a

Ziel der Übung: Erstellen von Service Effekt Spezifikationen mit stochastischen Ausdrücken im PCM

Allgemeines

Zu dieser Übung gibt es eine neue Übungsgruppeneinteilung, bei der sich nur zwei Gruppen, die Gruppen 10 und 8, verändert haben, damit niemand zu oft allein arbeiten muss. Es wurde zufällig bestimmt, wer einzeln arbeitet.

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Hinweis: Der Abgabetermin ist erst in zwei Wochen, dementsprechend hat das Übungsblatt 7 einen größeren Umfang. Dieser Teil entspricht daher der Übung für die erste Woche, spätestens am 13. Juni wird der zweite Teil herausgegeben.

Werkzeuge

Für die Bearbeitung der Aufgaben benötigen Sie die PCM Bench mit den aktuellsten Plugins (Führen Sie ein Update durch!) und die im Wiki verfügbare Musterlösung zu Übung 6.

Denken Sie unbedingt daran, bei dem Arbeiten mit der PCM Bench keine zwei Diagramme gleichzeitig geöffnet zu halten und zwischendurch öfter zu speichern. Falls Sie Probleme bei der Simulation haben (Plugin konnte nicht erzeugt werden), löschen Sie das automatisch generierte Projekt

Aufgaben

1. PCM: Sie haben das Informationssystem CBIS bereits in den Übungen 3, 5 und 6 bearbeitet. Nun sollen für die Abläufe, die in Übung 6 spezifiziert wurden, mithilfe stochastischer Ausdrücke nicht nur die Mittelwerte, sondern auch die Verteilungen angegeben werden, und außerdem die probabilitischen Verzweigungen durch von den Parametern abhängigen Verzweigungen ersetzt werden.

Die Wahrscheinlichkeiten für die Ausführung der einzelnen Anwendungsfälle müssen nur noch einmal im Usage Model definiert werden müssen, die Komponenten können unabhängig davon werden, indem die Verzweigungen und die Ressourcenanforderungen parametrisch angegeben werden.

Informationen zum Kontrollfluss:

Bisher wurde davon ausgegangen, dass bei der Generierung der Statistik im Schnitt 1000 Kunden geladen werden. Die genaue Anzahl der zu ladenden Kunden hängt davon ab, welche Parameter bei der Statistikgenerierung vom Benutzer angegeben werden. Sind die Parameter sehr einschränkend, müssen nur wenige Datensätze geladen werden. Sind die Parameter sehr weit gefasst oder liegen sogar gar keine vor, so werden viele oder sogar alle Kunden aus der Datenbank geladen. Im vorliegenden Fall wurde von Domänenexperten analysiert, welche Art von Anfragen die Benutzer wie häufig stellen. Wichtig für die Vorhersage der Performance ist es, wie viele Datensätze pro Anfrage benötigt werden. Daher wurden die Wahrscheinlichkeiten bestimmt, zu wie vielen Ergebnissen, d.h. zu wie viel ausgewählten Kunden, die Parameter führen. Weiterhin wird angenommen, dass die Datenbank 5500 Kundeneinträge hat.

Die Wahrscheinlichkeiten, zu wie vielen geladenen Kunden ein Parameter führt, sind als Verteilung angegeben, dabei gibt x die Anzahl der zu ladenden Kunden vor:

$P(0 <= x < 250) = 0.05$

$P(250 <= x < 750) = 0.35$

$P(750 <= x < 1250) = 0.2$

$P(1250 <= x < 2000) = 0.15$

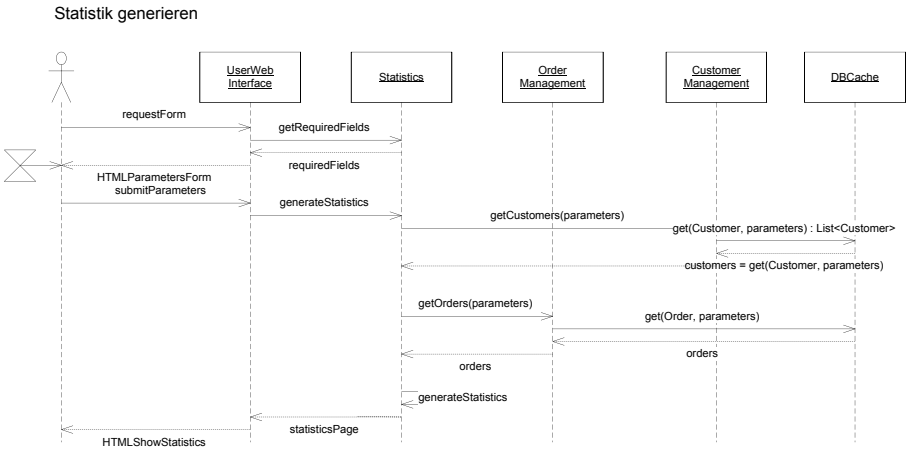
$P(2000 <= x < 3500) = 0.15$

$P(3500 <= x < 5500) = 0.1$

Sie müssen die Wahrscheinlichkeiten nicht für alle einzelnen möglichen Werte berechnen, sondern Sie können eine DoublePDF verwenden. So werden zwar in der Analyse zwar nicht nur jeweils eine ganzzahlige Anzahl von Kunden betrachtet, sondern auch Gleitkommazahlen verwendet, der dadurch entstehende Fehler sollte aber gering sein.

Wie viele Kunden für die Statistik geladen werden, hängt also von der Beschaffenheit der Parameter ab. Modellieren Sie diese Eigenschaft eines Parameters als STRUCTURE des Parameters. Die Parameter werden wiederum vom Benutzer bestimmt und sind in den an das UserWebInterface übergebenen formData enthalten, d.h. auch hier kann bereits die formData.STRUCTURE mit der obigen Verteilung angegeben werden. So wird erreicht, dass die Verteilung im Usage Model angegeben werden kann, und somit richtig modelliert, dass die Parameter von dem Benutzungsprofil bestimmt werden.

In den letzten Analysen wurde bereist deutlich, dass trotz der bereits verwendeten Verbesserungen immer noch das Generieren der Statistik sehr lange benötigt. Deshalb wurde dieser Anwendungsfall nochmals überarbeitet: Es werden nun alle passenden Aufträge in einem Batch geladen. Folgendes Sequenzdiagramm veranschaulicht den Ablauf.



Die Schleife zum Abrufen der Aufträge entfällt, alle Informationen sind in den Parametern enthalten und werden nun von der Datenbank direkt ausgelesen. Dies führt natürlich dazu, dass in diesem einen Zugriff viele Aufträge geladen werden. Weiterhin bleibt es dabei, dass im Mittel zwei Aufträge pro Kunde geladen werden.

Aufgrund dieser Änderung kann man aber nicht mehr davon ausgehen, dass 50% der Anfragen durch den Cache beantwortet werden können, da die Anfragen spezieller werden. Es können nun nur noch 20% der Anfragen aus dem Cache beantwortet werden.

Weiterhin wurde bisher davon ausgegangen, dass jeder Auftrag 10 Positionen enthält, die beim Anlegen des Auftrags einzeln in der Datenbank angelegt werden müssen. Nun ist auch hier eine genauere Verteilung für die Anzahl der Positionen y eines Auftrags bekannt:

$P(y = 7) = 0.1$

$P(y = 8) = 0.1$

$P(y = 9) = 0.2$

$P(y = 10) = 0.2$

$P(y = 11) = 0.2$

$P(y = 12) = 0.1$

$P(y = 13) = 0.1$

Ihre Aufgaben

1. Arbeiten Sie die oben angegebenen zusätzlichen Informationen in das CBIS Projekt ein. Verwenden Sie als Grundlage für Ihr Projekt die im Wiki verlinkte Musterlösung mit der Anpassung des FastStatistics-SEFF. Nennen Sie das Projekt um (Refactor->Rename), so dass es dem Schema „Uebung 07a Gruppe X“ entspricht.
2. Analysieren Sie das Modell erneut mit der Simulation für einen Benutzer, wie bereits in Übung 6. Simulieren Sie weiterhin nochmal die o.g. Grundlage ohne Anpassungen von Ihnen, indem Sie ein weiteres Projekt mit dieser Grundlage anlegen.

   Fügen Sie das Histogramm einem Dokument hinzu und geben Sie in 3 – 5 Sätzen an, was sich im Gegensatz zu den Ergebnissen aus Übung 6 verändert hat und woraus diese

Änderungen resultieren. Die Antwort sollte nicht zu knapp sein, sondern widerspiegeln, dass Sie Zusammenhänge erkennen. Sie braucht aber auch nicht übermäßig ausführlich sein und das Ergebnis bis ins Detail betrachten.

Hinweis: Wenn Sie mehrere Durchläufe machen, müssen Sie die Ansicht in der SimuBench aktualisieren, um auch die neusten Ergebnisse angezeigt zu bekommen.

3. Fragen zu den Konzepten:
   a) Benennen Sie die Unterschiede von Probability Mass Functions (Wahrscheinlichkeitsfunktion) und Probability Density Function (Dichtefunktion). Für welche Arten von Zufallsvariablen werden sie verwendet? Wie können Wahrscheinlichkeiten abgelesen werden?
   b) Beschreiben Sie mit einem eigenen Beispiel den Unterschied zwischen einer Schleife (Loop) und einem CollectionIterator.
4. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:
   1. Zeitangabe
   2. Zeitangabe
   3. Zeitangabe

Abgabe

Das PCM Projekt, das nach dem Schema „Uebung 07a Gruppe X“ benannt ist (Refactor->Rename), ein Dokument mit den Antworten zu Aufgabe 2 und 3 sowie die Angaben des Zeitaufwands müssen eingecheckt werden.

Ausgabe: 06.06.2007

Abgabe der Lösungen: 12.06.2007

Vorstellung der Musterlösung: 22.06.2007

Übungsblatt 7 b

Ziel der Übung: Erstellen eines kompletten Systems im PCM

Allgemeines

Zu dieser Übung gibt es eine neue Übungsgruppeneinteilung…

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Hinweis: Der Abgabetermin ist erst in zwei Wochen, dementsprechend hat das Übungsblatt 7 einen größeren Umfang. Dieser Teil entspricht daher der Übung für die erste Woche, spätestens am 13. Juni wird der zweite Teil herausgegeben.

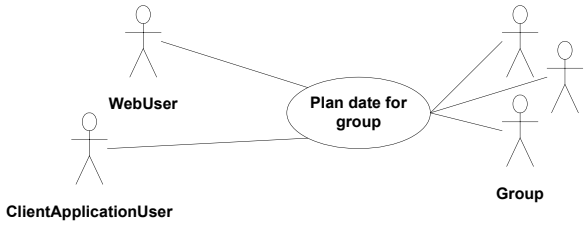
Werkzeuge

Für die Bearbeitung der Aufgaben benötigen Sie die PCM Bench mit den aktuellsten Plugins (Führen Sie ein Update durch!).

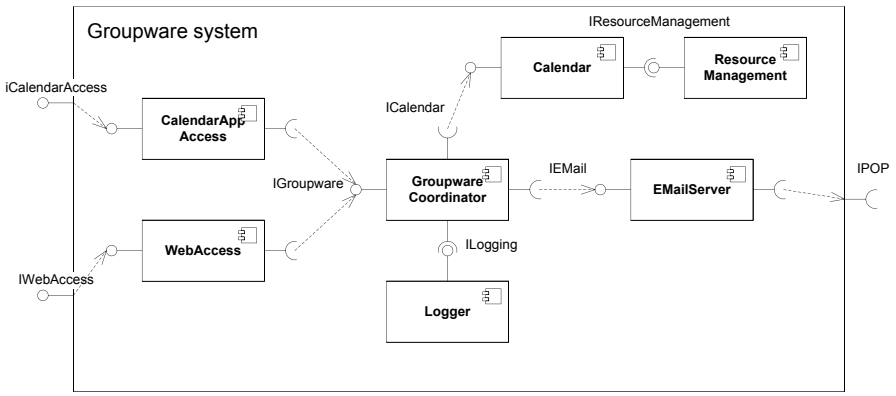
Denken Sie unbedingt daran, bei dem Arbeiten mit der PCM Bench keine zwei Diagramme gleichzeitig geöffnet zu halten und zwischendurch öfter zu speichern. Falls Sie Probleme bei der Simulation haben (Plugin konnte nicht erzeugt werden), löschen Sie das automatisch generierte Projekt

Aufgaben

Eine komponentenbasierte Groupware soll entwickelt werden, die sowohl über iCalendar Schnittstellen von verschiedener Kalendersoftware wie Mozilla Sunbird als auch über eine Weboberfläche verwendet werden können. Neben vielen weiteren Anwendungsfällen ist das Planen eines Termins für eine ganze Gruppe eine Funktion der Groupware und soll im Folgenden untersucht werden.



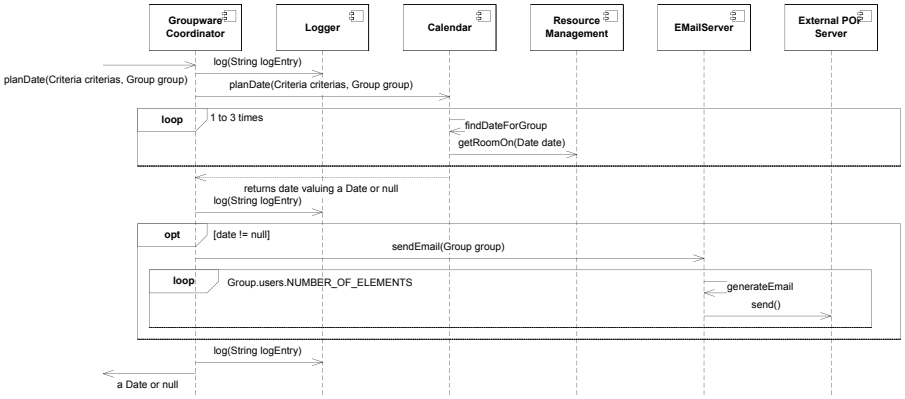
Das Groupware System ist wie folgt aufgebaut:



Alle Komponenten sind auf einem Server eingesetzt. Für die Festplatte des Servers ist die Latenzzeit weit größer als die zu lesenden oder schreibenden Daten, sie liegt bei 10 ms. Für die CPU wird die abstrakte Einheit der Work Units bei der Beschreibung der einzelnen Resource Demands verwendet. Die eingebaute CPU kann 1000 Work Units pro Sekunde verarbeiten.

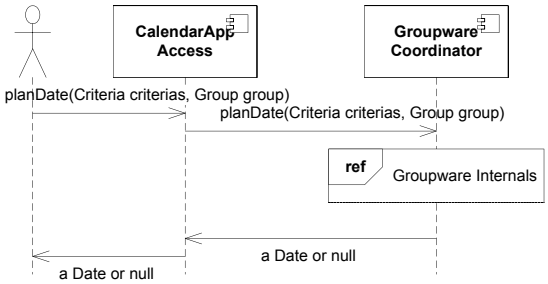
Das folgende Sequenzdiagramm beschreibt den Ablauf des Anwendungsfalls und beinhaltet gleichzeitig die benötigten Methoden der Signaturen. Falls die Methoden Rückgabewerte haben, sind diese durch einen Return-Pfeil gekennzeichnet, auf dem der Typ des Rückgabewerts vermerkt ist und die lokale Variable, der der Wert zugewiesen wird. Enthält ein späterer Methodenaufruf den Namen der lokalen Variablen, so bedeutet dies, dass diese lokale Variable übergeben wird.

Sequenzdiagramm Groupware Internals:

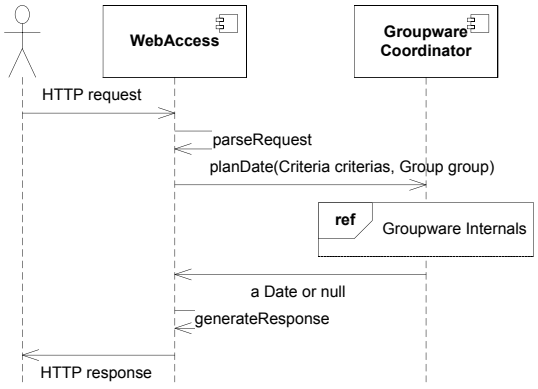


Der Aufruf der GroupwareCoordinator Komponente geschieht entweder über die iCalendarAccess Schnittstelle, die von verschiedener Kalendersoftware aufgerufen wird, oder über eine Webseite. Für beide Fälle folgen hier die Sequenzdiagramme.

Sequenzdiagramm iCalendar Access:



Sequenzdiagramm WebAccess



Benutzungsprofil:

Es wird geschätzt, dass alle 10 Sekunden ein Benutzer auf das System zugreift, dabei verwenden 30% die Weboberfläche und 70% andere Kalendersoftware. Die Gruppen, für die Termine gefunden werden sollen, haben ganz unterschiedliche Größen g, die durch folgende Verteilung beschrieben werden kann:

P(g = 3) = 0.05

P(g = 4) = 0.1

P(g = 5) = 0.1

P(g = 6) = 0.1

P(g = 7) = 0.15

P(g = 8) = 0.15

P(g = 9) = 0.15

P(g = 10) = 0.1

P(g = 11) =0.1

Resource Demands:

Das Parsen des HTTP Requests sowie das Generieren der HTTP Response benötigen jeweils 10 Work Units.

Das Schreiben eines Log-Eintrags benötigt einen Festplattenzugriff. Es gibt nur einen globalen Zeiger auf die Datei, auf den nur ein Prozess zu gleichen Zeit Zugriff haben kann (Acquire – Release).

Das Finden eines möglichen Termins für eine Gruppe (findDateForGroup) benötigt Rechenzeit der CPU und ist quadratisch abhängig von der Größe der Gruppe. Mit einem linearen Faktor ergibt sich die Formel (Größe der Gruppe)² * 10 Work Units.

Das Suchen nach einem Raum für einen gegebenen Termin benötigt 10 Work Units.

Für einen gefundenen Termin gibt es nicht unbedingt einen Raum. Die Komponente Calendar versucht bis zu dreimal, einen möglichen Termin und einen freien Raum zu finden. Falls kein Termin oder Raum gefunden werden kann, wird null zurückgegeben. Unabhängig davon wird das Ergebnis zu 80% im ersten Durchlauf gefunden und die Schleife wird abgebrochen, zu 15% erst im zweiten und zu 5% im dritten. Da es sowohl sein kann, dass kein Termin gefunden werden kann oder auch, dass für den dritten Termin auch kein Raum gefunden werden, nehmen wir an, dass die Anzahl der Schleifendurchläufe und die Tatsache, dass schlussendlich ein Termin gefunden wird, unabhängig sind. Zu 95% kann ein Termin gefunden werden, zu 5% nicht.

Die Komponente EMailServer generiert für jeden Benutzer in der Gruppe (auch den Terminsteller, falls dieser der Gruppe zugehört) eine Email und sendet sie über POP3. Das Generieren der Email benötigt 10 Work Units. Der Zeitverbrauch für das Senden einer Email kann als QoS Annotation für die Required-Schnittstelle des Systems modelliert werden. Der Zeitverbrauch hier ist eine halbe Sekunde.

Ansätze zur Verringerung der Antwortzeit

Zwei Entwurfsalternativen zur Verringerung der Antwortzeit werden vorgeschlagen.

Vorschlag 1

Zum einen könnte eine Komponente BufferedEMailServer eingekauft werden, die die Komponente EMailServer ersetzen kann. Diese Komponente speichert die generierten Emails erst einmal und verschickt sie erst nach der Rückgabe des Kontrollflusses über POP3. Es kann davon ausgegangen werden, dass alle anfallenden Emails irgendwann gesendet werden können, dass der Durchsatz hier also immer höher als die Ankunftsrate ist. Für die Modellierung bedeutet dies, dass der Aufruf von IPOP.send() gar nicht mehr modelliert werden muss.

Modellieren Sie die neue Komponente im Repository zusätzlich zur alten und legen Sie ein neues System, eine neue Allocation und ein neues Usage Model für diesen Fall an, so dass diese Alternative zusätzlich zum bestehenden System untersucht werden kann. Im Namen der neuen Modelle soll jeweils der String EA1 vorkommen, um sie vom Ausgangssystem unterscheiden zu können.

Vorschlag 2:

Der zweite Vorschlag ist recht einfach: Eine doppelt so schnelle CPU soll eingesetzt werden. Legen Sie hier ein neues Resource Environment, eine neue Allocation und ein neues Usage Model an. Im Namen der neuen Modelle soll jeweils der String EA2 vorkommen.

Hinweise

Um das System korrekt modellieren zu können, müssen Sie lokale Variablen verwenden.

Wie Lokale Variablen benutzen

Um Eclipse mit mehr Speicher zu starten, können Sie folgendes tun:

- Ändern Sie in der eclispe.ini Datei (im Eclipse Verzeichnis) den Parameter Xmx und geben Sie zum Beispiel -Xmx1024m an, um den maximal für den Java Heap zur Verfügung stehenden Speicher zu erhöhen. Der initial zur Verfügung stehende Java Heap wird mit –Xms angegeben.
- Sie können diese Parameter auch ebenso auf der Kommandozeile oder als Parameter einer Verknüpfung angeben.
  - Kommandozeile: eclipse.exe –Xmx1024m oder →

Ihre Aufgaben

1. Modellieren und analysieren Sie das System. Fügen Sie das Histogramm der Simulation in das Dokument Ihrer Abgabe ein. Was ist die ungefähre durchschnittliche Antwortzeit, was die maximale?
2. Untersuchen Sie die beiden weiteren Vorschläge und fügen Sie auch hier die Histogramme in die Abgabe ein. Geben Sie an, welche der mit dem Ausgangssystem drei Entwurfsalternativen die beste Antwortzeit ergibt und anhand welcher Kriterien Sie dies entschieden haben.
3. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:
   1. Zeitangabe
   2. Zeitangabe

Abgabe

Das PCM Projekt, das nach dem Schema „Uebung 07b Gruppe X“ benannt ist, ein Dokument mit den Antworten zu Aufgabe 1 und 2 sowie die Angaben des Zeitaufwands müssen eingecheckt werden.

Ausgabe: 15.06.2007

Abgabe der Lösungen: 26.06.2007

Vorstellung der Musterlösung: 29.06.2007

Übungsblatt 8a

Ziel der Übung 8: Erstellen eines kompletten Systems sowohl im SPE-ED als auch im PCM

Allgemeines

Zu dieser Übung gibt es eine neue Übungsgruppeneinteilung, die dann für den zweiten Teil des Übungsblatts größtenteils bestehen bleibt, nur eine Gruppe wird sich neu aufteilen müssen, damit niemand zweimal allein arbeiten muss.

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Hinweis: Der Abgabetermin ist erst in anderthalb Wochen.

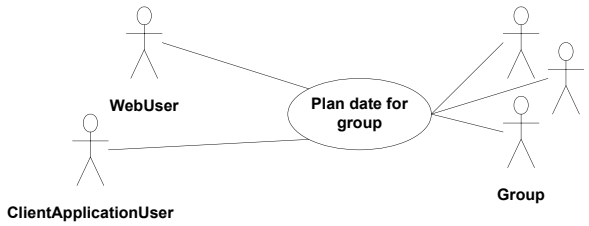
Werkzeuge

Sie benötigen für den ersten Teil der Übung das SPE-ED Werkzeug.

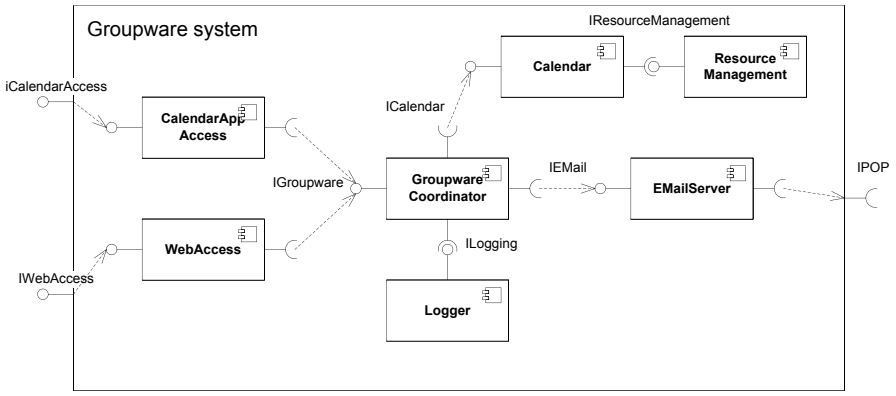
Aufgaben

Eine komponentenbasierte Groupware soll entwickelt werden, die sowohl über iCalendar Schnittstellen von verschiedener Kalendersoftware wie Mozilla Sunbird als auch über eine Weboberfläche verwendet werden können. Neben vielen weiteren Anwendungsfällen ist das Planen eines Termins für eine ganze Gruppe eine Funktion der Groupware und soll im Folgenden untersucht werden.

Für die Analyse der Antwortzeit soll jeweils die durchschnittliche sowie die maximale Antwortzeit untersucht werden.



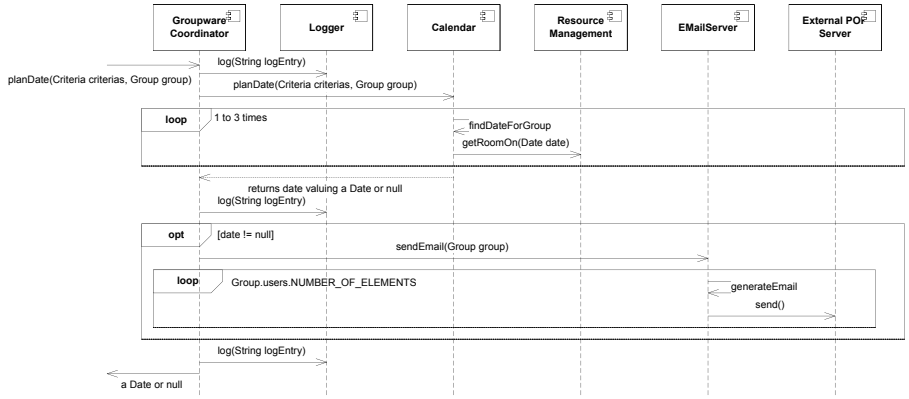
Das Groupware System ist wie folgt aufgebaut:



Alle Komponenten sind auf einem Server eingesetzt. Für die Festplatte des Servers ist die Latenzzeit weit größer als die zu lesenden oder schreibenden Daten, sie liegt bei 10 ms. Für die CPU wird die abstrakte Einheit der Work Units bei der Beschreibung der einzelnen Resource Demands verwendet. Die eingebaute CPU kann 1000 Work Units pro Sekunde verarbeiten.

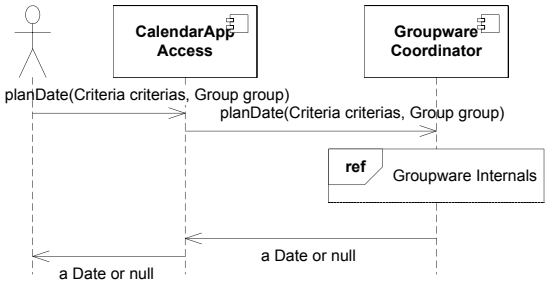
Das folgende Sequenzdiagramm beschreibt den Ablauf des Anwendungsfalls und beinhaltet gleichzeitig die benötigten Methoden der Signaturen. Falls die Methoden Rückgabewerte haben, sind diese durch einen Return-Pfeil gekennzeichnet, auf dem der Typ des Rückgabewerts vermerkt ist und die lokale Variable, der der Wert zugewiesen wird. Enthält ein späterer Methodenaufruf den Namen der lokalen Variablen, so bedeutet dies, dass diese lokale Variable übergeben wird.

Sequenzdiagramm Groupware Internals:

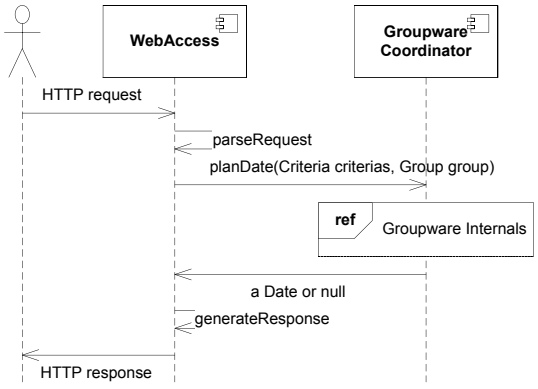


Der Aufruf der GroupwareCoordinator Komponente geschieht entweder über die iCalendarAccess Schnittstelle, die von verschiedener Kalendersoftware aufgerufen wird, oder über eine Webseite. Für beide Fälle folgen hier die Sequenzdiagramme.

Sequenzdiagramm iCalendar Access:



Sequenzdiagramm WebAccess



Benutzungsprofil:

Es wird geschätzt, dass alle 10 Sekunden ein Benutzer auf das System zugreift, dabei verwenden 30% die Weboberfläche und 70% andere Kalendersoftware. Die Gruppen, für die Termine gefunden werden sollen, haben ganz unterschiedliche Größen g, die durch folgende Verteilung beschrieben werden kann:

P(g = 3) = 0.05

P(g = 4) = 0.1

P(g = 5) = 0.1

P(g = 6) = 0.1

P(g = 7) = 0.15

P(g = 8) = 0.15

P(g = 9) = 0.15

P(g = 10) = 0.1

P(g = 11) =0.1

Resource Demands:

Das Parsen des HTTP Requests sowie das Generieren der HTTP Response benötigen jeweils 10 Work Units.

Das Schreiben eines Log-Eintrags benötigt einen Festplattenzugriff. Es gibt nur einen globalen Zeiger auf die Datei, auf den nur ein Prozess zu gleichen Zeit Zugriff haben kann (Passive Ressource, d.h. Acquire – Release). Dieses Verhalten kann in SPE nicht direkt modelliert werden. Geben Sie Lösungsideen an, wie man die Modellierung einer passiven Ressource prinzipiell gestalten könnte. Modellieren Sie hier auch entsprechend oder begründen Sie, warum der Einfluss vernachlässigt werden kann.

Das Finden eines möglichen Termins für eine Gruppe (findDateForGroup) benötigt Rechenzeit der CPU und ist quadratisch abhängig von der Größe der Gruppe. Mit einem linearen Faktor ergibt sich die Formel (Größe der Gruppe)² * 10 Work Units.

Das Suchen nach einem Raum für einen gegebenen Termin benötigt 10 Work Units.

Für einen gefundenen Termin gibt es nicht unbedingt einen Raum. Die Komponente Calendar versucht bis zu dreimal, einen möglichen Termin und einen freien Raum zu finden. Falls kein Termin oder Raum gefunden werden kann, wird null zurückgegeben. Unabhängig davon wird das Ergebnis zu 80% im ersten Durchlauf gefunden und die Schleife wird abgebrochen, zu 15% erst im zweiten und zu 5% im dritten. Da es sowohl sein kann, dass kein Termin gefunden werden kann oder auch, dass für den dritten Termin auch kein Raum gefunden werden, nehmen wir an, dass die Anzahl der Schleifendurchläufe und die Tatsache, dass schlussendlich ein Termin gefunden wird, unabhängig sind. Zu 95% kann ein Termin gefunden werden, zu 5% nicht.

Die Komponente EMailServer generiert für jeden Benutzer in der Gruppe (auch den Terminsteller, falls dieser der Gruppe zugehört) eine Email und sendet sie über POP3. Das Generieren der Email benötigt 10 Work Units. Der Zeitverbrauch für das Senden einer Email kann als QoS Annotation für die Required-Schnittstelle des Systems modelliert werden. Der Zeitverbrauch hier ist eine halbe Sekunde.

Ansätze zur Verringerung der Antwortzeit

Zwei Entwurfsalternativen zur Verringerung der Antwortzeit werden vorgeschlagen.

Vorschlag 1

Zum einen könnte eine Komponente BufferedEMailServer eingekauft werden, die die Komponente EMailServer ersetzen kann. Diese Komponente speichert die generierten Emails erst einmal und verschickt sie erst nach der Rückgabe des Kontrollflusses über POP3. Es kann davon ausgegangen werden, dass alle anfallenden Emails irgendwann gesendet werden können, dass der Durchsatz hier also immer höher als die Ankunftsrate ist. Für die Modellierung bedeutet dies, dass der Aufruf von IPOP.send() gar nicht mehr modelliert werden muss.

Modellieren Sie die neue Komponente im Repository zusätzlich zur alten und legen Sie ein neues System, eine neue Allocation und ein neues Usage Model für diesen Fall an, so dass diese Alternative zusätzlich zum bestehenden System untersucht werden kann. Im Namen der neuen Modelle soll jeweils der String EA1 vorkommen, um sie vom Ausgangssystem unterscheiden zu können.

Vorschlag 2:

Der zweite Vorschlag ist recht einfach: Eine doppelt so schnelle CPU soll eingesetzt werden. Legen Sie hier ein neues Resource Environment, eine neue Allocation und ein neues Usage Model an. Im Namen der neuen Modelle soll jeweils der String EA2 vorkommen.

Hinweise

Verwenden Sie globale Parameter, um Veränderungen leichter durchzuführen. Globale Parameter werden im SPEEDManual (nicht im Quickstart Guide, sondern im Wiki herunterzuladen) auf Seite 2-17 beschrieben.

Sie können ein modelliertes Szenario in einem weiteren Szenario wiederverwenden, dies bietet sich an, um für das Szenario WebAccess die Interna wiederzuverwenden. Vergleichen Sie dazu Seite 2-22 im SPEEDManual.

Sie benötigen keine weiteren Angaben von anderen Übungsblättern, für alles, was hier nicht aufgeführt wird, müssen keine Ressourcenanforderungen modelliert werden.

Ihre Aufgaben

1. Modellieren und analysieren Sie das System. Fügen Sie Screenshots eines durchschnittlichen Simulationslaufs in das Dokument Ihrer Abgabe ein. Was ist die ungefähre durchschnittliche Antwortzeit, was die maximale?
2. Untersuchen Sie die beiden weiteren Vorschläge und fügen Sie auch hier die Screenshots in die Abgabe ein. Geben Sie an, welche der mit dem Ausgangssystem drei Entwurfsalternativen das beste Antwortzeitverhalten ergibt und anhand welcher Kriterien Sie dies entschieden haben.
3. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:
    1. Zeitangabe
    2. Zeitangabe

Abgabe

Ein SPE-Verzeichnis namens „Uebung 08a Gruppe X“ mit dem Ausgangssystem sowie zwei SPE-ED Verzeichnisse, jeweils eines pro Entwurfsalternative EA, die nach dem Schema „Uebung 08a Gruppe X EA Y“ benannt ist, ein Dokument mit den Antworten zu Aufgabe 1 und 2 sowie die Angaben des Zeitaufwands müssen eingecheckt werden.

Ausgabe: 20.06.2007

Abgabe der Lösungen: 26.06.2007

Vorstellung der Musterlösung: 29.06.2007

Übungsblatt 8b

Ziel der Übung 8: Erstellen eines kompletten Systems sowohl im SPE-ED als auch im PCM

Allgemeines

Zu dieser Übung bleiben alle Übungsgruppen bis auf zwei bestehen. Nur die Gruppen 5 und 8 wurden neu aufgeteilt, damit niemand zweimal allein arbeiten muss.

Sollten Fragen oder Probleme bei der Bearbeitung des Übungsblattes auftreten, so steht Ihnen im SDQ-Wiki (`http://sdqweb.ipd.uka.de/wiki/`) ein Diskussionsbereich auf der Seite zum Praktikum zur Verfügung, in dem Fragen beantwortet werden.

Sollten darüber hinaus Probleme auftreten, wenden Sie sich bitte per E-Mail an martens@ipd.uka.de.

Die Abgabe der Lösungen erfolgt ebenso wie auf Übungsblatt 1 angegeben. Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben.

Hinweis: Das System ist dasselbe wie in Übung 8a. Es werden allerdings hier noch ein paar Hinweise zur Modellierung mit Palladio gemacht.

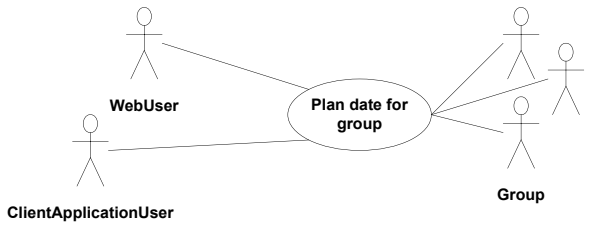
Werkzeuge

Sie benötigen für diesen zweiten Teil der Übung das PCM Werkzeug mit den aktuellsten Plugins.

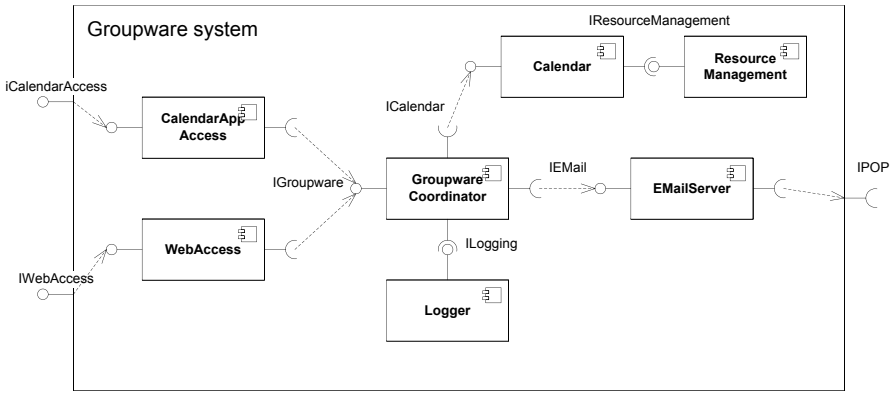
Aufgaben

Eine komponentenbasierte Groupware soll entwickelt werden, die sowohl über iCalendar Schnittstellen von verschiedener Kalendersoftware wie Mozilla Sunbird als auch über eine Weboberfläche verwendet werden können. Neben vielen weiteren Anwendungsfällen ist das Planen eines Termins für eine ganze Gruppe eine Funktion der Groupware und soll im Folgenden untersucht werden.

Für die Analyse der Antwortzeit soll jeweils die durchschnittliche sowie die maximale Antwortzeit untersucht werden.



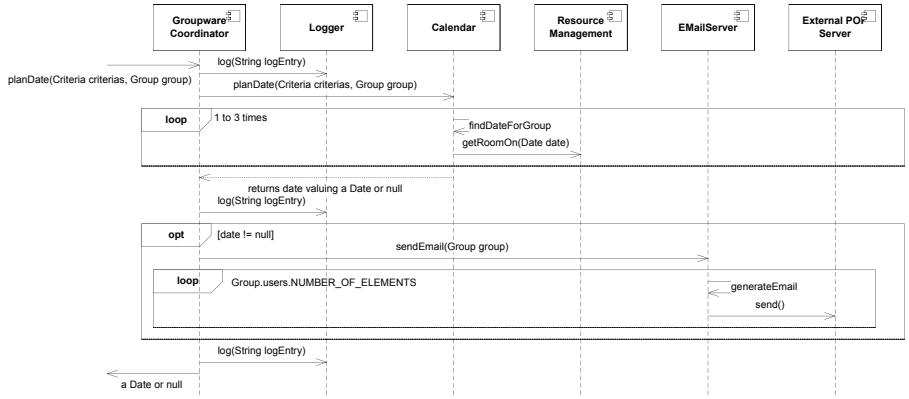
Das Groupware System ist wie folgt aufgebaut:



Alle Komponenten sind auf einem Server eingesetzt. Für die Festplatte des Servers ist die Latenzzeit weit größer als die zu lesenden oder schreibenden Daten, sie liegt bei 10 ms. Für die CPU wird die abstrakte Einheit der Work Units bei der Beschreibung der einzelnen Resource Demands verwendet. Die eingebaute CPU kann 1000 Work Units pro Sekunde verarbeiten.

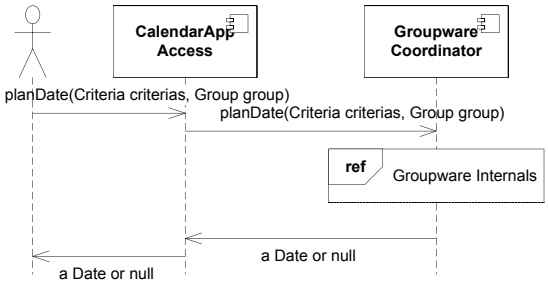
Das folgende Sequenzdiagramm beschreibt den Ablauf des Anwendungsfalls und beinhaltet gleichzeitig die benötigten Methoden der Signaturen. Falls die Methoden Rückgabewerte haben, sind diese durch einen Return-Pfeil gekennzeichnet, auf dem der Typ des Rückgabewerts vermerkt ist und die lokale Variable, der der Wert zugewiesen wird. Enthält ein späterer Methodenaufruf den Namen der lokalen Variablen, so bedeutet dies, dass diese lokale Variable übergeben wird.

Sequenzdiagramm Groupware Internals:

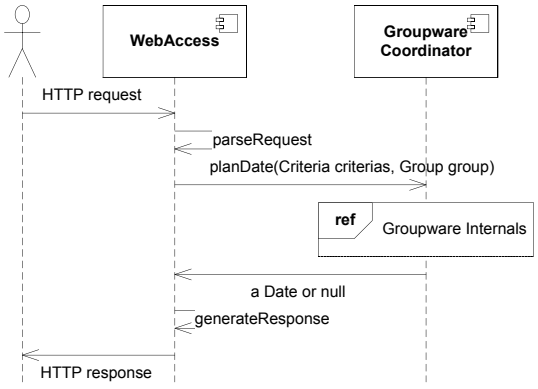


Der Aufruf der GroupwareCoordinator Komponente geschieht entweder über die iCalendarAccess Schnittstelle, die von verschiedener Kalendersoftware aufgerufen wird, oder über eine Webseite. Für beide Fälle folgen hier die Sequenzdiagramme.

Sequenzdiagramm iCalendar Access:



Sequenzdiagramm WebAccess



Benutzungsprofil:

Es wird geschätzt, dass alle 10 Sekunden ein Benutzer auf das System zugreift, dabei verwenden 30% die Weboberfläche und 70% andere Kalendersoftware. Die Gruppen, für die Termine gefunden werden sollen, haben ganz unterschiedliche Größen g, die durch folgende Verteilung beschrieben werden kann:

P(g = 3) = 0.05

P(g = 4) = 0.1

P(g = 5) = 0.1

P(g = 6) = 0.1

P(g = 7) = 0.15

P(g = 8) = 0.15

P(g = 9) = 0.15

P(g = 10) = 0.1

P(g = 11) =0.1

Resource Demands:

Das Parsen des HTTP Requests sowie das Generieren der HTTP Response benötigen jeweils 10 Work Units.

Das Schreiben eines Log-Eintrags benötigt einen Festplattenzugriff. Es gibt nur einen globalen Zeiger auf die Datei, auf den nur ein Prozess zu gleichen Zeit Zugriff haben kann (Passive Ressource, d.h. Acquire – Release). Dieses Verhalten kann in SPE nicht direkt modelliert werden. Geben Sie Lösungsideen an, wie man die Modellierung einer passiven Ressource prinzipiell gestalten könnte. Modellieren Sie hier auch entsprechend oder begründen Sie, warum der Einfluss vernachlässigt werden kann.

Das Finden eines möglichen Termins für eine Gruppe (findDateForGroup) benötigt Rechenzeit der CPU und ist quadratisch abhängig von der Größe der Gruppe. Mit einem linearen Faktor ergibt sich die Formel (Größe der Gruppe)² * 10 Work Units.

Das Suchen nach einem Raum für einen gegebenen Termin benötigt 10 Work Units.

Für einen gefundenen Termin gibt es nicht unbedingt einen Raum. Die Komponente Calendar versucht bis zu dreimal, einen möglichen Termin und einen freien Raum zu finden. Falls kein Termin oder Raum gefunden werden kann, wird null zurückgegeben. Unabhängig davon wird das Ergebnis zu 80% im ersten Durchlauf gefunden und die Schleife wird abgebrochen, zu 15% erst im zweiten und zu 5% im dritten. Da es sowohl sein kann, dass kein Termin gefunden werden kann oder auch, dass für den dritten Termin auch kein Raum gefunden werden, nehmen wir an, dass die Anzahl der Schleifendurchläufe und die Tatsache, dass schlussendlich ein Termin gefunden wird, unabhängig sind. Zu 95% kann ein Termin gefunden werden, zu 5% nicht.

Die Komponente EMailServer generiert für jeden Benutzer in der Gruppe (auch den Terminsteller, falls dieser der Gruppe zugehört) eine Email und sendet sie über POP3. Das Generieren der Email benötigt 10 Work Units. Der Zeitverbrauch für das Senden einer Email kann als QoS Annotation für die Required-Schnittstelle des Systems modelliert werden. Der Zeitverbrauch hier ist eine halbe Sekunde.

Ansätze zur Verringerung der Antwortzeit

Zwei Entwurfsalternativen zur Verringerung der Antwortzeit werden vorgeschlagen.

Vorschlag 1

Zum einen könnte eine Komponente BufferedEMailServer eingekauft werden, die die Komponente EMailServer ersetzen kann. Diese Komponente speichert die generierten Emails erst einmal und verschickt sie erst nach der Rückgabe des Kontrollflusses über POP3. Es kann davon ausgegangen werden, dass alle anfallenden Emails irgendwann gesendet werden können, dass der Durchsatz hier also immer höher als die Ankunftsrate ist. Für die Modellierung bedeutet dies, dass der Aufruf von IPOP.send() gar nicht mehr modelliert werden muss.

Modellieren Sie die neue Komponente im Repository zusätzlich zur alten und legen Sie ein neues System und eine neue Allocation für diesen Fall an, so dass diese Alternative zusätzlich zum bestehenden System untersucht werden kann. Im Namen der neuen Modelle soll jeweils der String EA1 vorkommen, um sie vom Ausgangssystem unterscheiden zu können.

Sie können das bestehende System auch kopieren und nur die eine Komponente austauschen. Allerdings löscht der Composite Editor für eine gelöschte Assembly noch nicht die Konnektoren. Sie müssen in diesem Fall also die beiden Konnektoren, die noch die Assembly_EMailServer referenzieren, im Baumeditor von Hand löschen.

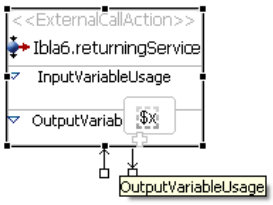
Vorschlag 2:

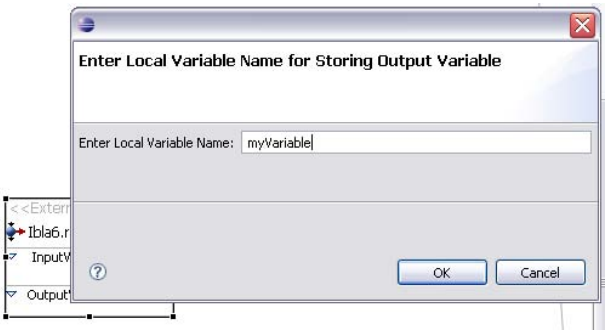
Der zweite Vorschlag ist recht einfach: Eine doppelt so schnelle CPU soll eingesetzt werden. Legen Sie hier ein neues Resource Environment und eine neue Allocation an. Im Namen der neuen Modelle soll jeweils der String EA2 vorkommen.

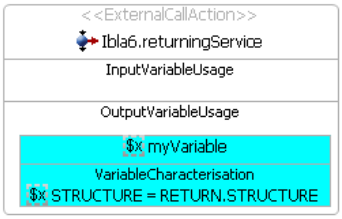
Hinweise

Um das System korrekt modellieren zu können, müssen Sie lokale Variablen verwenden.

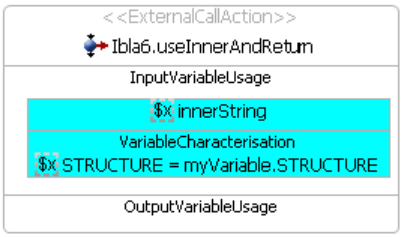
Wenn Sie die Output Variable Usage eines External Calls anlegen, werden Sie nach dem Namen einer lokalen Variablen gefragt. Sie können einen neuen Namen eingeben, von nun an steht eine Variable mit diesem Namen zur Verfügung. Sie können ihm dann den von der Methode zurückgegebenen Wert zuweisen. Sie können ihn dann in der weiteren Parametrisierung wie einen Parameter der aktuellen Methode verwenden (s.u.).





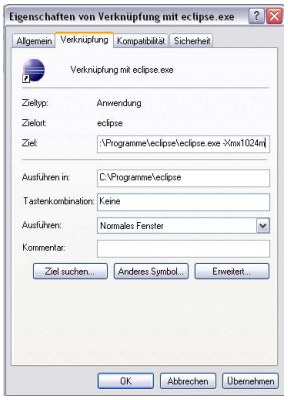


Verwendung im weiteren Verlauf, beispielsweise als Eingabeparameter in einem weiteren External Call:



Um Eclipse mit mehr Speicher zu starten, können Sie folgendes tun:

- Ändern Sie in der eclispe.ini Datei (im Eclipse Verzeichnis) den Parameter Xmx und geben Sie zum Beispiel -Xmx1024m an, um den maximal für den Java Heap zur Verfügung stehenden Speicher zu erhöhen. Der initial zur Verfügung stehende Java Heap wird mit –Xms angegeben.
- Sie können diese Parameter auch ebenso auf der Kommandozeile oder als Parameter einer Verknüpfung angeben.
  - Kommandozeile: eclipse.exe –Xmx1024m oder →



Ihre Aufgaben

1. Modellieren und analysieren Sie das System. Fügen Sie die CDF eines Simulationslaufs in das Dokument Ihrer Abgabe ein. Was ist die ungefähre durchschnittliche Antwortzeit, was die maximale? Hinweis: In der Vorlesung am Freitag wird u.a. die Interpretation der Ergebnisse behandelt.
2. Untersuchen Sie die beiden weiteren Vorschläge und fügen Sie auch hier die CDFs in die Abgabe ein. Geben Sie an, welche der insgesamt drei Entwürfe das beste Antwortzeitverhalten ergibt, welcher das zweit beste und welcher das schlechteste, und anhand welcher Kriterien Sie dies entschieden haben.
3. Zeitaufwand: Bitte geben Sie in einer separaten Textdatei „Zeitaufwand.txt“ an, wie viel Zeit Sie für die Bearbeitung der einzelnen Aufgaben benötigt haben:
   1. Zeitangabe
   2. Zeitangabe

Abgabe

Das PCM Projekt, das nach dem Schema „Uebung 08b Gruppe X“ benannt ist, ein Dokument mit den Antworten zu Aufgabe 1 und 2 (CDFs im Dokument! Also kein reines Textformat) sowie die Angaben des Zeitaufwands müssen eingecheckt werden.

A.6 Durations of Solving Preparatory Exercises

Table A.1 shows the durations for solving the preparatory exercises for the single participants (linewise) and in average (last line). Missing values were not provided by the participant

Exercise 2		Ex. 3	Ex. 4	Exercise 5			Ex. 6	Ex. 7	Ex. 8a	Ex. 8b
SPE	Pal	SPE	SPE	Pal	SPE		Pal	Pal	SPE	Pal
50	40	220	210	45	30	120	300	0	420	960
60	60	240	330	120	60	60	165	360	300	960
45	30	420	240	90	60	30	300	360	170	180
45	30	300	360	90	90	120	240	330	300	420
70	90	170	420	130	180	60	300	210	102	300
120	90	600	420	120	180	20	360	240	105	300
60	45	300	360	90	90	120	540	210	210	330
60	45		420	130	180	60	480	225	180	300
30	180	100	240	90	60	30		360	390	960
30	10	100	260	45	60	30	210	120	360	660
60	45	240	210	120	30	45	360	300	420	960
50	40	220	210	120	90	10	165	120	210	330
		100	0	120	120	30	230	120	180	
30	30	170	420	120	180	20	480	170	170	180
90	60	300	330	120	60	60	360	170	180	480
90	120	420	360	105	40	15	300	365	270	270
60	30	300	360	105	40	15	195	300	270	480
60	30	100	210	120	90	10	180	240	390	960
		360	260	45	60	30		365	660	360
59.4	57.4	258.9	312.2	101.3	89.5	46.6	303.8	253.6	278.3	532.4

Table A.1: Durations of solving preparatory exercises

B Experimental Material

B.1 Experiment Tasks

The reference models can be found on the accompanying DVD of this thesis. If you are reading a printed version without DVD or an electronic version of this work, and are interested in the reference models, please contact the author.

B.1.1 Media Store

Palladio

Palladio

Fallstudie: Vergleich von Performanzvorhersageverfahren

Ein neues komponentenbasiertes System zur Speicherung und Abfrage von MP3-Dateien im Internet soll entwickelt werden. Während der gesamten Entwicklung soll die Performanz des Entwurfs bewertet werden, um den späteren Benutzern ein System mit akzeptablen Antwortzeiten bieten zu können. Da Sie mit der Performanzanalyse vertraut sind, werden Sie gebeten, verschiedene Entwurfsalternativen hinsichtlich ihrer Performanz zu untersuchen.

Zeitstempel

Um untersuchen zu können, wie viel der Zeit auf welche Aufgabenteile entfällt, geben Sie bitte jeweils minutengenaue Zeiten für die Teilaufgaben an. Sie müssen dabei die Aufgabenteile nicht alle in der Reihenfolge abarbeiten, in der Sie sie hier vorfinden. Differenzieren Sie bei den Zeitstempeln weiterhin nach der Zeit für die reine Modellierung und der Zeit für die Fehlersuche.

Modellieren und analysieren Sie aber zunächst das Ausgangsystem vollständig und erst danach die Entwurfsalternativen, wenn Ihr Ausgangssystem abgenommen wurde. Führen Sie auch bei den Entwurfsalternativen erst die Analyse einer Entwurfsalternative durch, bevor Sie zur nächsten Übergehen. Geben Sie Ihre Ergebnisse direkt in das Ergebnisblatt am Ende ein.

Checken Sie Ihre Lösung jeweils zu einem Zeitstempel in Ihren SVN Account ein, um ein Backup zu haben.

Zeitstempel auf Extrablatt angeben: Austeilung der Aufgabenstellung.

Übersicht über die Aufgabenstellung

Durchführung der Untersuchungen

Modellieren und analysieren Sie zunächst das Ausgangsystem. Lassen Sie Ihre Modellierung und die Ergebnisse überprüfen, bevor Sie mit der Modellierung der Entwurfsalternativen beginnen.

Modellieren und analysieren Sie die Entwurfsalternativen in der hier angegebenen Reihenfolge. Analysieren Sie eine fertiggestellte Entwurfsalternative und tragen Sie die Ergebnisse in die Tabelle am Ende ein, bevor Sie mit der nächsten Entwurfsalternative fortfahren.

Arbeiten Sie in einem Verzeichnis Mediastore-Nachname, das Sie bei jedem Zeitstempel (oder auch häufiger) in Ihr SVN Repository einchecken. Geben Sie in der Log Message den Zeitstempel an.

Palladio

Untersuchen Sie mit dem PCM die Antwortzeit des Systems. Modellieren Sie hierfür die Parameter nur mit den für die Performanz relevanten Charakterisierungen. Nutzen Sie die vorhandenen Parameter der Signaturen im Repository für Charakterisierungen und verwenden Sie auch Komponentenparameter, wenn die Information nicht aus den Signaturenparametern geschlossen werden kann.

Wichtig: Laden Sie Modelle nur über „Browse Workspace“, da sie sonst absolut verlinkt sind und nicht auf anderen Systemen geöffnet werden können.

Legen Sie für die Entwurfsalternativen, bei denen andere Komponenten eingesetzt werden, ein neues System und eine neue Allokation an, die diese neue Komponente verwenden. Sie können dazu das alte System kopieren und anpassen. Die Allokation muss neu angelegt werden. Benennen Sie die neuen Modelle mit dem Suffix -EA-x, wobei x die Nummer der Alternative angibt. Sie erhalten beispielsweise das System Model File mediaStore-EA-1.system.

SEFFs müssen neu modelliert werden, sie können nicht aus anderen Komponenten kopiert werden.

Falls nur das Resource Environment und/oder die Allokation verändert wird, brauchen Sie nur diese Modelle neu zu erstellen. Auch hier kann das Resource Environment kopiert und angepasst werden. Verwenden Sie hier ebenfalls das oben angegebene Namensschema.

Legen Sie ein zweites Usage Model für das zweite Benutzungsprofil an. Für die Entwurfsalternative 2 benötigen Sie weiterhin zwei eigene Benutzungsprofile mit den angepassten Dateigrößen.

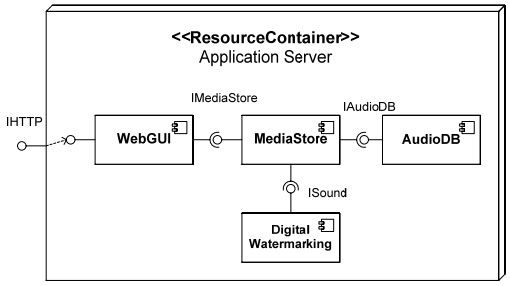
Erzeugen Sie weiterhin für jede Entwurfsalternative und jedes Benutzungsprofil eine eigene Run Config, die Sie ebenfalls nach dem obigen Schema benennen. Sie haben am Ende also 12 Run Configs angelegt, jeweils zwei pro Entwurfsalternative, weil pro Benutzungsprofil eines. Benennen Sie die Run Configs und die Simulationsergebnisse wie oben angegeben plus ein Suffix BP-y für das Benutzungsprofil y.

Sie können die verschiedenen Analysen nicht parallel ausführen, sondern eine nach der anderen.

Speichern Sie die Analyseergebnisse für das Ausgangssystem und die einzelnen Entwurfsalternativen als Grafikdatei mit demselben Namensschema. Legen Sie weiterhin auch die Datenbank mit den Simulationsergebnissen im Projektverzeichnis ab.

Bestehender Entwurf

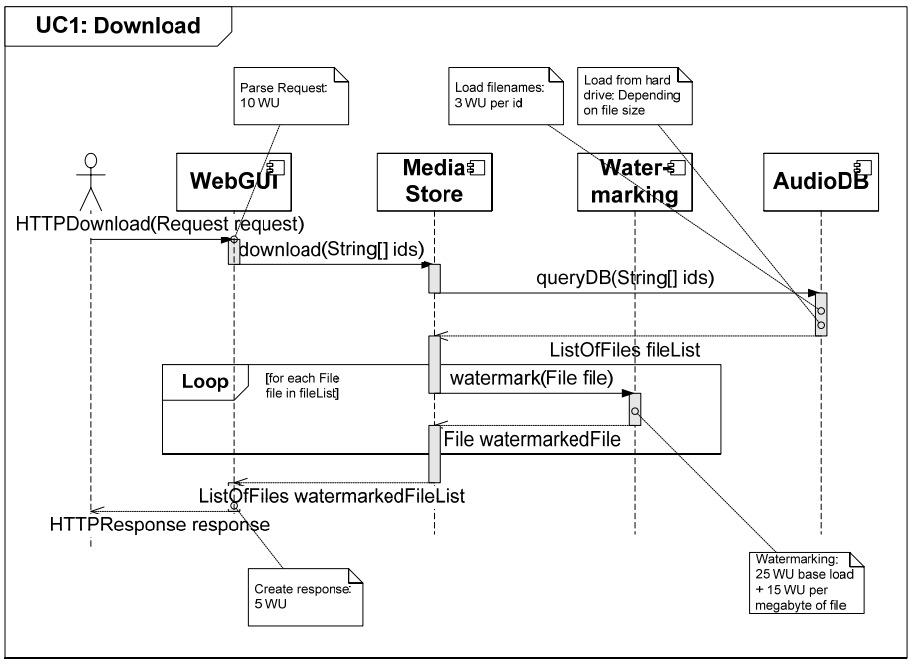
Das System besteht aus den folgenden Komponenten, die alle auf einem Applikationsserver eingesetzt sind:



Das dazugehörige Repository mit den benötigten Komponenten, den Schnittstellen und ihren Signaturen wurden bereits im PCM modelliert. Laden Sie das Projekt aus dem Wiki[1] herunter, benennen Sie es nach dem Schema PCM_Mediastore_Nachname um und vervollständigen Sie es mit den folgenden Angaben.

Die beiden entscheidenden Anwendungsfälle sind das Herunterladen und Hochladen von Musikstücken im MP3-Format.

Anwendungsfall 1: Herunterladen:



[1] http://sdqweb.ipd.uka.de/wiki/Praktikum_Ingenieurmäßiger_Software-Entwurf_SS07

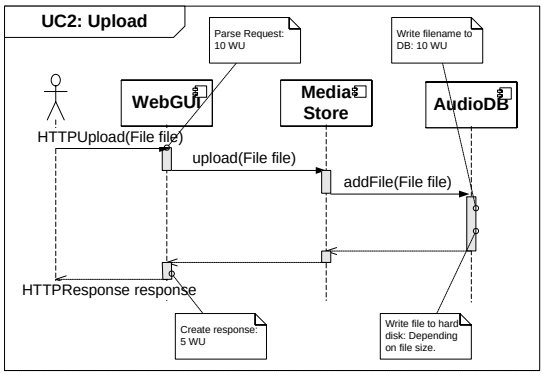
Über die `WebGUI` wird das System vom Benutzer verwendet. Im HTTP Request wird eine Liste von IDs der angeforderten Musikstücke an die `WebGUI` übergeben. Das Parsen des HTTP Requests benötigt 10 Work Units. Die Anfrage wird danach vom `MediaStore` an die `AudioDB` weitergeleitet.

Die Komponente `AudioDB` liefert die gewünschten Musikstücke zurück, indem sie zunächst in einer Schleife für jede gegebene ID den Dateinamen aus einer speziellen internen Datenbank im Hauptspeicher lädt (Aufwand pro ID konstant 3 Work Units) und dann die Datei selbst von der Festplatte lädt (vgl. Angaben für die Festplatte auf Seite 7). Die Liste der geladenen Musikstücke wird an die `MediaStore` Komponente zurückgegeben.

Von dort aus wird jedes einzelne Musikstück an die Komponente `Watermarking` übergeben, die ein digitales Wasserzeichen einarbeitet und dann das Musikstück zurückgibt. Die Größe des Musikstücks bleibt unverändert. Das Hinzufügen des Wasserzeichens benötigt konstant 25 Work Units und weitere 15 Work Units pro zu verarbeitendem MB (1 MB entspricht 10^6 Byte).

Das Erzeugen des HTTP Response benötigt weitere 5 Work Units.

Anwendungsfall 2: Hochladen



Es werden nur einzelne Musikstücke hochgeladen. Aus dem HTTP Request an die `WebGUI` wird das hochzuladende Musikstück entnommen (Aufwand zum Parsen 10 Work Units) und an die `MediaStore` Komponente weitergereicht. Die `MediaStore` Komponente leitet das Musikstück an die `AudioDB` Komponente weiter. Hier wird zunächst ein Eintrag in die interne Datenbank gemacht (Aufwand konstant 10 Work Units) und dann das Musikstück in eine Datei auf die Festplatte geschrieben (vgl. Angaben für die Festplatte auf Seite 7).

Das Erzeugen des HTTP Response benötigt weitere 5 Work Units.

Zeitstempel auf Extrablatt angeben: Kontrollfluss und Resource Demand

Benutzungsprofile

Domänenexperten wurden zu Rate gezogen, um das spätere Benutzungsprofil zu ermitteln. Sie kamen zu diesen Schlussfolgerungen:

Benutzungsprofil 1

- Es wird immer ein Benutzer das System verwenden. Dies entspricht einem Closed Workload mit einem Benutzer und einer Think Time von 0.
- In 20% der Fälle werden MP3 Dateien hochgeladen (UC2: Upload), in 80% der Fälle werden sie heruntergeladen (UC1: Download).
- Die Dateigröße x in Megabytes der Musikstücke ist sehr unterschiedlich, sie können durch die folgende Verteilung angenähert werden:

$$P(0{,}5\ MB <= x < 1\ MB) = 0.051$$
$$P(1\ MB <= x < 2\ MB) = 0.134$$
$$P(2\ MB <= x < 3\ MB) = 0.193$$
$$P(3\ MB <= x < 4\ MB) = 0.212$$
$$P(4\ MB <= x < 5\ MB) = 0.224$$
$$P(5\ MB <= x < 6\ MB) = 0.186$$

  Der Mittelwert liegt bei einer Dateigröße von 3,5 MB. Die Verteilung ist sowohl für den Download als auch für den Upload relevant.

  Modellieren Sie die Größe der Dateien als Komponentenparameter.

- Die Anzahl der heruntergeladenen Stücke n liegt zwischen einem und 12 Stücken (ein Album). Um die Spezifikation zu vereinfachen, wurden einige Größenbereiche zu einem zusammengefasst. Verwenden Sie zur Spezifikation der Verteilung jeweils den Mittelwert eines jeden Bereichs.

$$P(n \in \{1,2,3\}) = 0.2$$
$$P(n \in \{4,5,6\}) = 0.2$$
$$P(n \in \{7,8,9\}) = 0.3$$
$$P(n \in \{10,11,12\}) = 0.3$$

Nur für Entwurfsalternative 4 relevant, aber trotzdem Teil des Benutzungsprofils:

- MP3 Dateien können mit variabler oder konstanter Bitrate kodiert sein. Hier werden 30 % der Musikstücke mit konstanter Bitrate hochgeladen, 70 % mit variabler.
- Die durchschnittliche Bitrate b der hochgeladenen Musikstücke (egal ob konstant oder variabel) ist wie folgt verteilt:

$$P(b = 64\ kbps) = 0.1$$
$$P(b = 128\ kbps) = 0.5$$
$$P(b = 192\ kbps) = 0.2$$
$$P(b = 320\ kbps) = 0.2$$

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 1

Benutzungsprofil 2

Es werden weiterhin Vermutungen über die Veränderungen des Benutzungsprofils mit der Zeit angestellt, denn schließlich soll das System auch dann noch akzeptable Antwortzeiten haben.

- Nach der Verbreitung des Systems werden auch mehrere Benutzer das System parallel nutzen. Es wird eine Zwischenankunftszeit von 1,25 Sekunden erwartet.
- Die Anzahl der heruntergeladenen Dateien wird sich erhöhen, da immer mehr Dateien verfügbar sind. Nehmen Sie an, dass sich die Verteilung um 1 nach oben verschiebt. Um die Spezifikation zu vereinfachen, wurden einige Größenbereiche zu einem zusammengefasst. Verwenden Sie zur Spezifikation der Verteilung jeweils den Mittelwert eines jeden Bereichs. Sie können auch 1 auf die alte Verteilung addieren.

  $P(n \in \{2,3,4\}) = 0.2$
  $P(n \in \{5,6,7\}) = 0.2$
  $P(n \in \{8,9,10\}) = 0.3$
  $P(n \in \{11,12,13\}) = 0.3$
- Das Verhältnis der Anwendungsfälle verschiebt sich weiter in Richtung UC1:Download, 90% der Benutzer laden Dateien herunter, nur 10% laden Dateien hoch, da bereits ein reicher Fundus an Musikstücken vorrätig sein wird.
- Die Bitraten sowie die Dateigröße der Musikstücke bleiben so wie in Benutzungsprofil 1.

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 2

Ressourcenumgebung

Alle Komponenten sind auf einem Server eingesetzt, wie das obige Diagramm bereits zeigt. Die CPU des Systems ist ein AMD Athlon XP mit einer Taktrate von 1,145 GHz = 1,145 * 10^9 Zyklen/s. Ein CPU Zyklus entspricht einer Instruktion. Für die folgenden Angaben wird der Rechenaufwand in Work Units angegeben. Ein Work Unit entspricht ca. 1,145 * 10^6 Instruktionen = 1,145 * 10^6 CPU Zyklen. Es wurde gemessen, dass die Festplatte im Durchschnitt 24 MB/s = 24 * 10^6 Byte/s Lesen und Schreiben kann. Ein MB kann also in 42 ms gelesen bzw. geschrieben werden. Die Latenz (Seek Time) der Festplatte kann vernachlässigt werden.

Das Resource Environment ist für Palladio bereits vorgegeben. Sie müssen allerdings noch die Verarbeitungsgeschwindigkeiten der Geräte angeben. Für die CPU bietet sich z.B. als Einheit „Work Units / s“ an, für die Festplatte „Byte/s“ an. Beide Geräte haben bereits die benötigte Scheduling Policy „Processor Sharing“ voreingestellt.

Zeitstempel auf Extrablatt angeben: Ressourcenumgebung

Entwurfsalternativen

Modellieren Sie die Entwurfsalternativen erst, wenn das Ausgangsystem vollständig modelliert ist. Lassen Sie Ihr Ausgangssystem erst überprüfen, bevor Sie die Entwurfsalternativen modellieren.

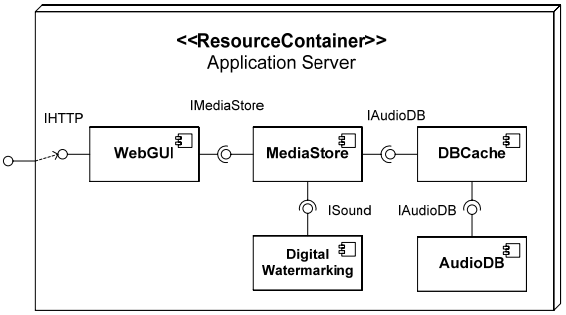
Entwurfsalternative 1: Cache

Motivation

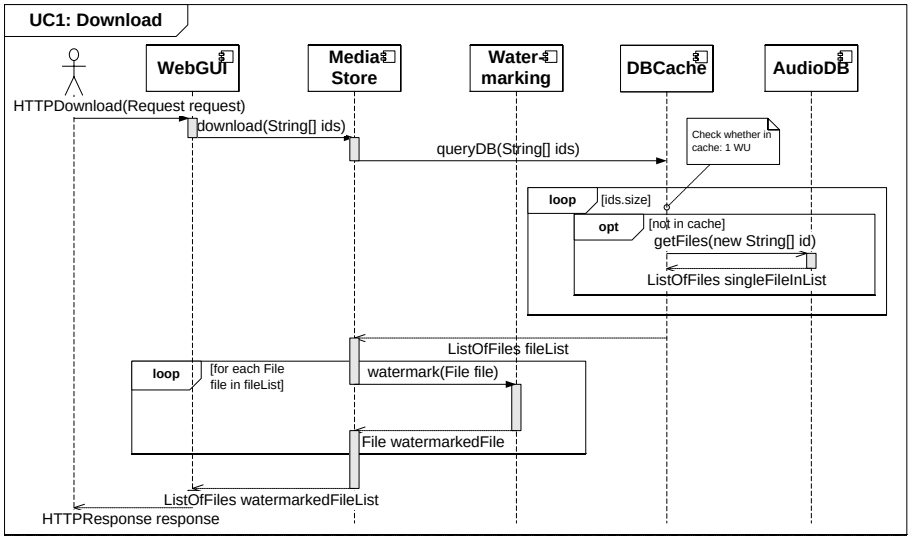
Es soll ein Cache eingesetzt werden, der einige Musikstücke zwischenspeichert und somit in manchen Fällen den Datenbankzugriff einsparen kann. Allerdings benötigt das Überprüfen des Caches selbst wieder ein wenig Rechenleistung.

Anpassung des Systems

Eine zusätzliche Komponente `DBCache` wird in das System eingeführt. Die folgenden Diagramme zeigen das angepasste System und den angepassten Kontrollfluss.



Ablauf mit zusätzlichem Resource Demand (nicht dargestellte Resource Demands bleiben unverändert):



Die Komponente `DBCache` erhält vom `MediaStore` die Liste aller angeforderten Musikstücke. Für jedes Musikstück wird geprüft, ob es im Cache vorliegt. Falls nicht, wird es einzeln von der Datenbank angefordert. `AudioDB.getFiles` wird also pro Schleifendurchlauf nur mit der einen jeweiligen id in der Liste aufgerufen und liefert eine Liste mit nur dem einen entsprechenden Musikstück zurück. Durch die Schleife wird dann die gesamte Liste zusammengestellt.

Cache Hit Ratio

Obwohl der Cache nur wenige Musikstücke im Verhältnis zur Gesamtgröße der Datenbank fassen kann, wird vermutet, dass ein Anteil der angefragten Musikstücke bereits im Cache vorgehalten sein wird. Die Strategie des Caches wird dazu auch Informationen zu den aktuellen Charts verwenden, da beliebte, aktuelle Musikstücke viel häufiger nachgefragt werden und somit im Cache vorgehalten werden können. Es wird davon ausgegangen, dass so eine Cache Hit Ratio von 20 % erreicht werden kann.

Rechenaufwand zum Prüfen des Caches

Das Prüfen, ob ein Musikstück im Cache vorliegt, kann dank eines Indexes mit nur 1 Work Units erledigt werden.

Palladio: Sie müssen die Verteilung der Dateigrößen nun auch nochmal als Komponentenparameter an den Cache hängen, damit sie auch für Dateien aus dem Cache angegeben werden kann.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 1

Entwurfsalternative 2: Pool für Datenbankverbindungen

Motivation

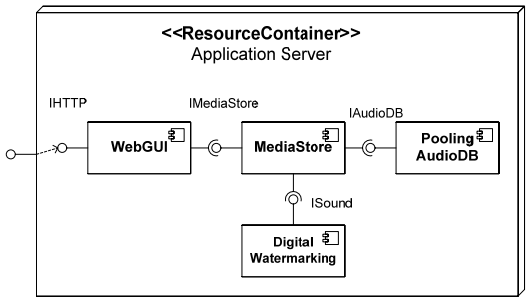
Um die Zeit für den Zugriff auf die interne Datenbank der `AudioDB` Komponente zu senken, kann ein Pool für Datenbankverbindungen eingeführt werden. Dabei muss dann nicht für jede Anfrage an die Datenbank eine neue Datenbankverbindung geöffnet werden, sondern es wird eine Datenbankverbindung aus dem Pool verwendet und nach der Benutzung wieder freigegeben. Allerdings kann so nur noch eine gewisse Anzahl von Threads gleichzeitig auf die Datenbank zugreifen.

Anpassung des Systems

In diesem Fall soll ein Pool der Größe 5 eingeführt werden. Die bestehende Komponente `AudioDB` wird ersetzt durch die Komponente `PoolingAudioDB`, die diese 5 Datenbankverbindungen verwaltet. Zusätzlich zu dem oben beschrieben Verhalten wird vor dem Zugriff auf die interne Datenbank der `AudioDB` Komponente eine Datenbankverbindung reserviert, und nach dem Datenbankzugriff wieder freigegeben.

Angaben zum Rechenaufwand

Dadurch verringert sich der Rechenaufwand für die Abfrage der Datenbank auf 1 Work Unit, für das Schreiben in die Datenbank (Use Case Upload) auf 5 Work Units.



Zeitstempel auf Extrablatt angeben: Entwurfsalternative 2

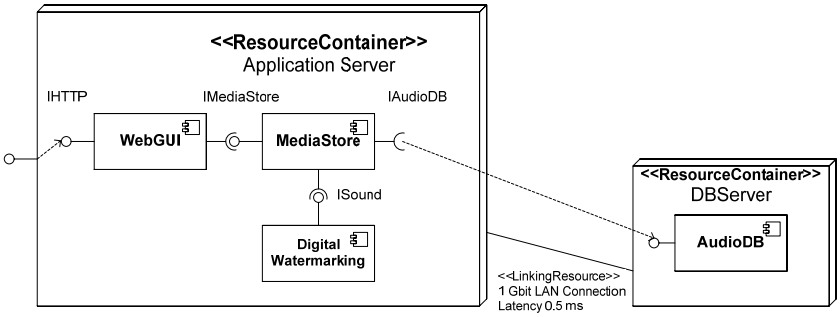
Entwurfsalternative 3: Zweiten Server für die Datenbankkomponenten

Motivation

Es wird ein zweiter Server bereitgestellt, auf den die AudioDB ausgelagert werden soll.

Anpassung des Systems

Die folgenden Diagramme zeigen das angepasste Deployment, der Kontrollfluss und das System verändern sich nicht.



Angaben zur Ressourcenumgebung

Der Datenbankserver hat dieselben Leistungsmerkmale wie der Applikationsserver.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 3

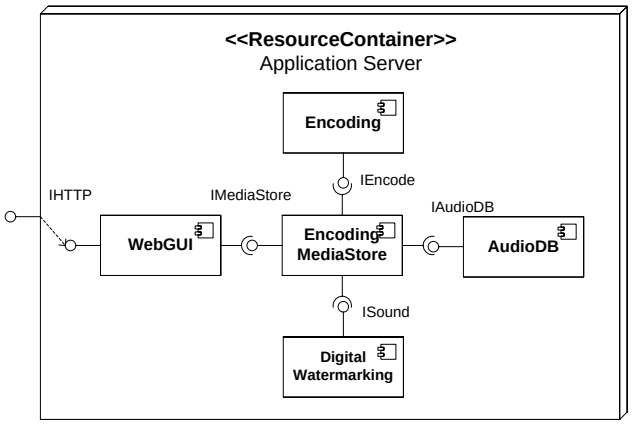
Entwurfsalternative 4: Bitrate senken

Motivation

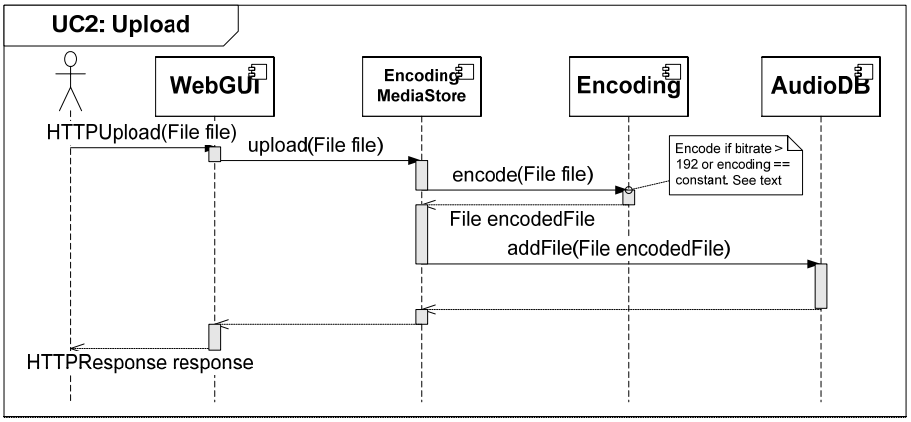
Musikstücke werden in unterschiedlicher Qualität hochgeladen. Weiterhin bietet das MP3 Format die Möglichkeit, Musikstücke mit konstanter oder variabler Bitrate zu kodieren, wobei bei der Verwendung einer variablen Bitrate in den meisten Fällen kleinere Dateien derselben Qualität erzeugt werden können. Um Speicherplatz und Zeit beim Herunterladen zu sparen, kann die Bitrate beim Hochladen verändert werden: Zum einen können für Musikstücke mit hohen durchschnittlichen Bitraten die durchschnittliche Bitrate gesenkt werden, wodurch allerdings die Qualität der Stücke sinkt. Weiterhin können Musikstücke, die mit einer konstanten Bitrate erzeugt wurden, unter Verwendung einer variablen Bitrate umgewandelt werden. Dabei sinkt die Qualität kaum.

Anpassung des Systems

Eine zusätzliche Komponente `Encoding` wird in die Architektur eingeführt. Die folgenden Diagramme zeigen das angepasste System und den angepassten Kontrollfluss für den Upload Anwendungsfall.



Ablauf mit zusätzlichem Resource Demand (nicht dargestellte Resource Demands bleiben unverändert):



Die Komponente `Encoding` komprimiert ein Musikstück nicht immer, sondern nur, wenn die Bitrate a) konstant ist oder b) höher als 192 kbps ist. Beide Fälle können zusammen auftreten, die Effekte werden kombiniert.

- Wenn die Bitrate konstant ist, kann durch die Umwandlung in eine mit variabler Bitrate kodierte MP3 Datei die Größe der Datei auf durchschnittlich 70% ihrer ursprünglichen Größe reduziert werden.
- Wenn die Bitrate höher als 192 kbps, wird sie auf 192 kbps gesenkt. Es kann davon ausgegangen werden, dass die Größe der Datei im selben Verhältnis abnimmt wie die Bitrate.

Wenn beide Fälle zusammen auftreten, wird trotzdem das Musikstück nur einmal neu kodiert, d.h. es fällt nur einmal der Rechenaufwand an. Die Größe des Musikstücks wird dann um 70% und um das Verhältnis der neuen zur alten Bitrate abnehmen.

Rechenaufwand zum Neukodieren (für Upload)

Wenn ein Musikstück neu kodiert werden soll, dekodiert die `Encoding` Komponente das Musikstück zunächst in das .wav Format, was einen Rechenaufwand von 10 Work Units pro MB der Ausgangsgröße bedeutet. Danach wird die Datei mit der neuen Bitrate kodiert, dies benötigt 10 Work Units pro MB der resultierenden MP3.

Falls die Bitrate des Musikstücks nicht geändert wird, wenn also die Bitrate variabel und nicht größer als 192 kbps ist, fällt kein Rechenaufwand an.

Veränderter Speicherbedarf (für Download)

Diese Rekodierung wirkt sich natürlich auch entsprechend auf die Dateigrößen der herunterzuladenen Musikstücke aus. Im Durchschnitt werden die Musikstücke also für die beiden gegebenen Benutzungsprofile um ca. 17% kleiner. Man erhält die folgende Verteilungsfunktion für die Dateigröße x in MB:

$$P(0{,}42 <= x < 0{,}84) = 0.051$$

$$P(0{,}84 <= x < 1{,}67) = 0.134$$

$$P(1{,}67 <= x < 2{,}5) = 0.193$$

$$P(2{,}5 <= x < 3{,}35) = 0.212$$

$$P(3{,}35 <= x < 4{,}2) = 0.224$$

$$P(4{,}2 <= x < 5) = 0.186$$

Verwenden Sie die Verteilungsfunktion nur als Hilfe bei der Modellierung des Ladens von der Festplatte. Für die Modellierung des Rechenaufwands der Rekodierung können diese Angaben nicht verwendet werden, denn hier werden nicht alle Stücke neu kodiert. Es müssen für die Neukodierung also die Angaben der Aufzählung oben verwendet werden.

Geben Sie die neue Verteilung als Komponentenparameter der `AudioDB` Komponente im System an.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 4

Entwurfsalternative 5: Dynamischer Lookup

Motivation

Diese Entwurfsalternative hat im Gegensatz zu den bisherigen Alternativen nicht zum Ziel, die Performanz des Systems zu verbessern, sondern die Skalierbarkeit. Wenn alle Komponenten ihr Kommunikationspartner über einen dynamischen Lookup beim Broker finden, kann die Allokation der Komponenten leicht verändert werden, beispielsweise können neue Server angeschafft und einige der Komponenten darauf eingesetzt werden, ohne Änderungen am Code vornehmen zu müssen. Allerdings benötigt das dynamische Auffinden des Kommunikationspartners zusätzlichen Rechenaufwand.

Es wurde beschlossen, dass der dynamische Lookup nur verwendet werden soll, wenn sein Einsatz die Antwortzeit des Systems für die gegebenen Benutzungsprofile um nicht mehr als 10% verschlechtert. Analysieren Sie, ob dieses Performanzziel eingehalten werden kann und geben Sie an, ob diese Entwurfsalternative nach diesen Vorgaben eingesetzt werden kann.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 5

Fragestellungen

Untersuchen Sie das Ausgangssystem und die verschiedenen Entwurfsalternativen und sichern Sie die Ergebnisse wie oben unter Durchführung (Seite 2) angegeben.

1. Welche Entwurfsalternativen sollten eingesetzt werden*, welche nicht? Differenzieren Sie Ihre Antwort nach den beiden Benutzungsprofilen. Geben Sie Ihre Antwort und eine Begründung in der Tabelle 1 unten an.
   (*: Die Alternative 5 braucht die Antwortzeit nicht verbessern, darf sie aber nicht mehr als 10% verschlechtern, um eingesetzt zu werden.)
2. Angenommen, es wäre nicht möglich, alle sinnvollen Entwurfsalternativen umzusetzen, sondern nur einige, welche wäre das? Stellen Sie eine Reihenfolge der Entwurfsentscheidungen auf, um diese Frage zu beantworten, und sortieren Sie die Entwurfsentscheidungen in Tabelle 2, beginnend mit der nützlichsten. Beachten Sie dabei nicht eventuelle Abhängigkeiten der Entwurfsentscheidungen untereinander.

Entwurfsalternative	Benutz-ungs-profil	Einsetzen? Mit Begründung.
1: Cache	1	
	2	
2: Pool für Datenbank-verbindungen	1	
	2	
3: Zweiter Server	1	
	2	
4: Bitrate senken	1	
	2	
5: Dynamischer Lookup	1	
	2	

Tabelle 1: Bewertung der Entwurfsalternativen

	Reihenfolge	
Entwurfsalternative	**Benutzungsprofil 1**	**Benutzungsprofil 2**
1: Cache		
2: Pool für Datenbankverbindungen		
3: Zweiter Server		
4: Bitrate senken		

Tabelle 2: Reihenfolge der Entwurfsalternativen

Kommentare zu den Entwurfsalternativen:

Zusatzfrage:

Bitte bearbeiten Sie diese Frage erst, wenn alle anderen Fragen beantwortet und die Antworten akzeptiert worden sind.

1. Wenn Sie beliebig viele Entwurfsalternativen umsetzen können, welche Kombination ergibt die besten Antwortzeiten?

Für diese Fragestellung müssen Sie verschiedene Kombinationen von Entwurfsalternativen zusammen analysieren und bewerten.

Beste Kombination:

Begründung:

SPE

SPE-ED

Fallstudie: Vergleich von Performanzvorhersageverfahren

Ein neues komponentenbasiertes System zur Speicherung und Abfrage von MP3-Dateien im Internet soll entwickelt werden. Während der gesamten Entwicklung soll die Performanz des Entwurfs bewertet werden, um den späteren Benutzern ein System mit akzeptablen Antwortzeiten bieten zu können. Da Sie mit der Performanzanalyse vertraut sind, werden Sie gebeten, verschiedene Entwurfsalternativen hinsichtlich ihrer Performanz zu untersuchen.

Zeitstempel

Um untersuchen zu können, wie viel der Zeit auf welche Aufgabenteile entfällt, geben Sie bitte jeweils minutengenaue Zeiten für die Teilaufgaben an. Sie müssen dabei die Aufgabenteile nicht alle in der Reihenfolge abarbeiten, in der Sie sie hier vorfinden. Differenzieren Sie bei den Zeitstempeln weiterhin nach der Zeit für die reine Modellierung und der Zeit für die Fehlersuche.

Modellieren und analysieren Sie aber zunächst das Ausgangsystem vollständig und erst danach die Entwurfsalternativen, wenn Ihr Ausgangssystem abgenommen wurde. Führen Sie auch bei den Entwurfsalternativen erst die Analyse einer Entwurfsalternative durch, bevor Sie zur nächsten Übergehen. Geben Sie Ihre Ergebnisse direkt in das Ergebnisblatt am Ende ein.

Checken Sie Ihre Lösung jeweils zu einem Zeitstempel in Ihren SVN Account ein, um ein Backup zu haben.

Zeitstempel auf Extrablatt angeben: Austeilung der Aufgabenstellung.

Übersicht über die Aufgabenstellung

Durchführung der Untersuchungen

Modellieren und analysieren Sie zunächst das Ausgangsystem. Lassen Sie Ihre Modellierung und die Ergebnisse überprüfen, bevor Sie mit der Modellierung der Entwurfsalternativen beginnen.

Modellieren und analysieren Sie die Entwurfsalternativen in der hier angegebenen Reihenfolge. Analysieren Sie eine fertiggestellte Entwurfsalternative und tragen Sie die Ergebnisse in die Tabelle am Ende ein, bevor Sie mit der nächsten Entwurfsalternative fortfahren.

Arbeiten Sie in einem Verzeichnis Mediastore-Nachname, das Sie bei jedem Zeitstempel (oder auch häufiger) in Ihr SVN Repository einchecken. Geben Sie in der Log Message den Zeitstempel an.

SPE-ED

Untersuchen Sie mit SPE-ED die durchschnittliche Antwortzeit.

Legen Sie für jede Entwurfsalternative (EA) und Benutzungsprofil ein eigenes Projekt an. Sie können die Projekte auch in unterschiedlichen Verzeichnissen in eigenen SPE-ED Installationen ablegen, damit Sie das Ausgangsprojekt kopieren und weiterverwenden können. Nennen Sie entweder die Projekte MediaStore-EA-x-BP-y, wobei x die Nummer der Alternative (0 für den Ausgangsentwurf) und y die Nummer des Benutzungsprofils angibt, oder benennen Sie die Verzeichnisse nach diesem Schema.

Wichtig: Bedenken Sie, dass Sie nur vier Navigationsboxen zur Verfügung haben und Sie nicht zu viele Expand Nodes einsetzen können. Gehen Sie also sparsam mit Expand Nodes um, indem Sie Sequenzen innerhalb einer Expand Node zusammenfassen und als eine Basic Node modellieren. Nur für Schleifen und Verzweigungen sollten Expand Nodes verwendet werden, aber auch hier nicht immer. Überlegen Sie, wie Sie die verfügbaren Navigationsboxen verwenden, bevor Sie mit der Modellierung beginnen.

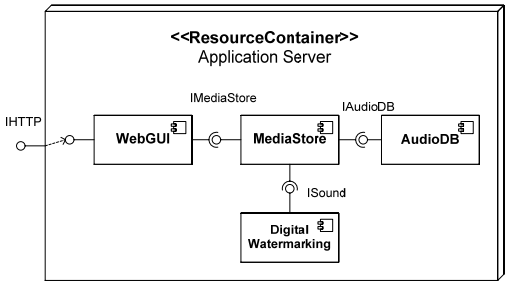
Bewerten Sie die Entwurfsalternativen aufgrund der Antwortzeiten für beide Anwendungsfälle. Der einfachste Weg hierfür ist es, jeweils beide Anwendungsfälle in einem Szenario zu modellieren. Sie haben dann den Vorteil, dass Sie nur die „Scenario Contention Analyse“ durchführen müssen. Für Entwurfsalternative 4 benötigen Sie ein weiteres Szenario auf einer weiteren Facility. Untersuchen Sie die Antwortzeit für dieses Szenario und setzen Sie das Ergebnis in einen Synchronisationsknoten im Ausgangsszenario als Delay ein. Auch hier benötigen Sie also keine Simulation.

Prüfen Sie bei jeder Analyse, ob immer noch die richtige Analyseart ausgewählt ist.

Speichern Sie einen Screenshot der Analyseergebnisse für das Ausgangssystem und die einzelnen Entwurfsalternativen als Grafikdatei mit demselben Namensschema. Insgesamt müssen Sie 12 Dateien speichern, durch die Kombination von 6 Entwurfsalternativen (inklusive dem Ausgangssystem als EA0) mit zwei Benutzungsprofilen.

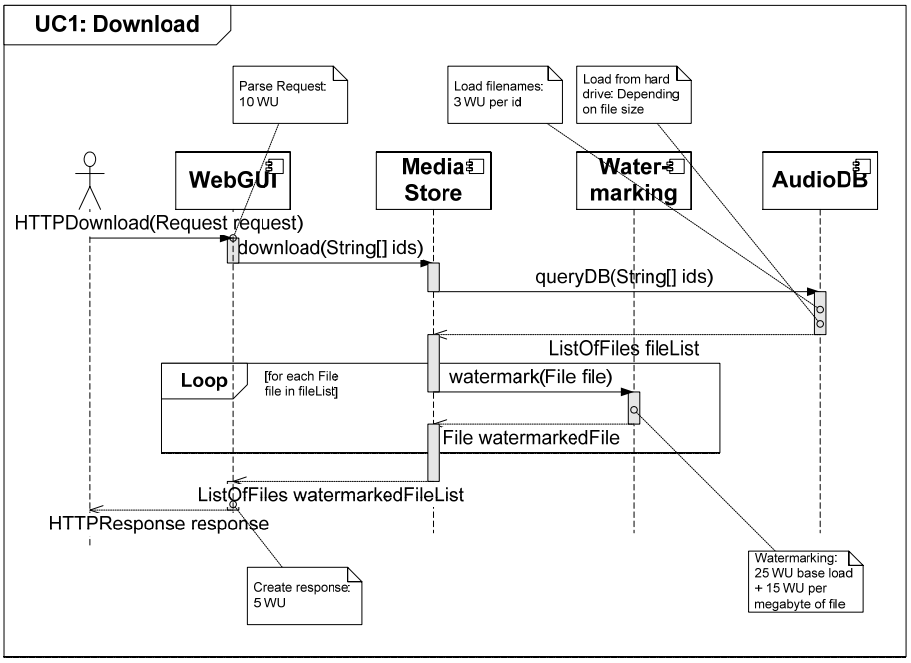
Bestehender Entwurf

Das System besteht aus den folgenden Komponenten, die alle auf einem Applikationsserver eingesetzt sind:



Die beiden entscheidenden Anwendungsfälle sind das Herunterladen und Hochladen von Musikstücken im MP3-Format.

Anwendungsfall 1: Herunterladen:



Über die `WebGUI` wird das System vom Benutzer verwendet. Im HTTP Request wird eine Liste von IDs der angeforderten Musikstücke an die `WebGUI` übergeben. Das Parsen des HTTP Requests benötigt 10 Work Units. Die Anfrage wird danach vom `MediaStore` an die `AudioDB` weitergeleitet.

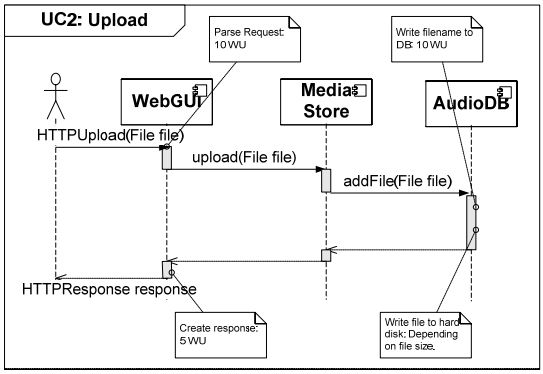
Die Komponente `AudioDB` liefert die gewünschten Musikstücke zurück, indem sie zunächst in einer Schleife für jede gegebene ID den Dateinamen aus einer speziellen internen Datenbank im Hauptspeicher lädt (Aufwand pro ID konstant 3 Work Units) und dann die

Datei selbst von der Festplatte lädt (vgl. Angaben für die Festplatte auf Seite 7). Die Liste der geladenen Musikstücke wird an die `MediaStore` Komponente zurückgegeben.

Von dort aus wird jedes einzelne Musikstück an die Komponente `Watermarking` übergeben, die ein digitales Wasserzeichen einarbeitet und dann das Musikstück zurückgibt. Die Größe des Musikstücks bleibt unverändert. Das Hinzufügen des Wasserzeichens benötigt konstant 25 Work Units und weitere 15 Work Units pro zu verarbeitendem MB (1 MB entspricht 10^6 Byte).

Das Erzeugen des HTTP Response benötigt weitere 5 Work Units.

Anwendungsfall 2: Hochladen



Es werden nur einzelne Musikstücke hochgeladen. Aus dem HTTP Request an die `WebGUI` wird das hochzuladende Musikstück entnommen (Aufwand zum Parsen 10 Work Units) und an die `MediaStore` Komponente weitergereicht. Die `MediaStore` Komponente leitet das Musikstück an die `AudioDB` Komponente weiter. Hier wird zunächst ein Eintrag in die interne Datenbank gemacht (Aufwand konstant 10 Work Units) und dann das Musikstück in eine Datei auf die Festplatte geschrieben (vgl. Angaben für die Festplatte auf Seite 7).

Das Erzeugen des HTTP Response benötigt weitere 5 Work Units.

Modellieren Sie für SPE-ED beide Anwendungsfälle in einem Szenario. So können Sie eine durchschnittliche Antwortzeit für beide Anwendungsfälle ermitteln.

Zeitstempel auf Extrablatt angeben: Kontrollfluss und Resource Demand

Benutzungsprofile

Domänenexperten wurden zu Rate gezogen, um das spätere Benutzungsprofil zu ermitteln. Sie kamen zu diesen Schlussfolgerungen:

Benutzungsprofil 1

- Es wird immer ein Benutzer das System verwenden. Dies entspricht einem Closed Workload mit einem Benutzer und einer Think Time von 0.
- In 20% der Fälle werden MP3 Dateien hochgeladen (UC2: Upload), in 80% der Fälle werden sie heruntergeladen (UC1: Download).
- Die Dateigröße x in Megabytes der Musikstücke ist sehr unterschiedlich, sie können durch die folgende Verteilung angenähert werden:

$$P(0{,}5\ MB <= x < 1\ MB) = 0.051$$
$$P(1\ MB <= x < 2\ MB) = 0.134$$
$$P(2\ MB <= x < 3\ MB) = 0.193$$
$$P(3\ MB <= x < 4\ MB) = 0.212$$
$$P(4\ MB <= x < 5\ MB) = 0.224$$
$$P(5\ MB <= x < 6\ MB) = 0.186$$

  Der Mittelwert liegt bei einer Dateigröße von 3,5 MB. Die Verteilung ist sowohl für den Download als auch für den Upload relevant.

- Die Anzahl der heruntergeladenen Stücke n liegt zwischen einem und 12 Stücken (ein Album). Um die Spezifikation zu vereinfachen, wurden einige Größenbereiche zu einem zusammengefasst. Verwenden Sie zur Spezifikation der Verteilung jeweils den Mittelwert eines jeden Bereichs.

$$P(n \in \{1,2,3\}) = 0.2$$
$$P(n \in \{4,5,6\}) = 0.2$$
$$P(n \in \{7,8,9\}) = 0.3$$
$$P(n \in \{10,11,12\}) = 0.3$$

  Der gesamte Mittelwert liegt damit bei 7,1 Musikstücken.

Nur für Entwurfsalternative 4 relevant, aber trotzdem Teil des Benutzungsprofils:

- MP3 Dateien können mit variabler oder konstanter Bitrate kodiert sein. Hier werden 30 % der Musikstücke mit konstanter Bitrate hochgeladen, 70 % mit variabler.
- Die durchschnittliche Bitrate b der hochgeladenen Musikstücke (egal ob konstant oder variabel) ist wie folgt verteilt:

$$P(b = 64\ kbps) = 0.1$$
$$P(b = 128\ kbps) = 0.5$$
$$P(b = 192\ kbps) = 0.2$$
$$P(b = 320\ kbps) = 0.2$$

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 1

Benutzungsprofil 2

Es werden weiterhin Vermutungen über die Veränderungen des Benutzungsprofils mit der Zeit angestellt, denn schließlich soll das System auch dann noch akzeptable Antwortzeiten haben.

- Nach der Verbreitung des Systems werden auch mehrere Benutzer das System parallel nutzen. Es wird eine Ankunftsrate von 0,8 Benutzern pro Sekunde erwartet.
- Die Anzahl der heruntergeladenen Dateien wird sich erhöhen, da immer mehr Dateien verfügbar sind. Nehmen Sie an, dass sich die Verteilung um 1 nach oben verschiebt. Um die Spezifikation zu vereinfachen, wurden einige Größenbereiche zu einem zusammengefasst. Verwenden Sie zur Spezifikation der Verteilung jeweils den Mittelwert eines jeden Bereichs. Sie können bei Palladio auch 1 auf die alte Verteilung addieren.

  $$P(n \in \{2,3,4\}) = 0.2$$
  $$P(n \in \{5,6,7\}) = 0.2$$
  $$P(n \in \{8,9,10\}) = 0.3$$
  $$P(n \in \{11,12,13\}) = 0.3$$

  Der gesamte Mittelwert liegt damit bei 8,1 Musikstücken.
- Das Verhältnis der Anwendungsfälle verschiebt sich weiter in Richtung UC1:Download, 90% der Benutzer laden Dateien herunter, nur 10% laden Dateien hoch, da bereits ein reicher Fundus an Musikstücken vorrätig sein wird.
- Die Bitraten sowie die Dateigröße der Musikstücke bleiben so wie in Benutzungsprofil 1.

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 2

Ressourcenumgebung

Alle Komponenten sind auf einem Server eingesetzt, wie das obige Diagramm bereits zeigt. Die CPU des Systems ist ein AMD Athlon XP mit einer Taktrate von 1,145 GHz = 1,145 * 10^9 Zyklen/s. Ein CPU Zyklus entspricht einer Instruktion. Für die folgenden Angaben wird der Rechenaufwand in Work Units angegeben. Ein Work Unit entspricht ca. 1,145 * 10^6 Instruktionen = 1,145 * 10^6 CPU Zyklen. Es wurde gemessen, dass die Festplatte im Durchschnitt 24 MB/s = 24 * 10^6 Byte/s Lesen und Schreiben kann. Ein MB kann also in 42 ms gelesen bzw. geschrieben werden. Die Latenz (Seek Time) der Festplatte kann vernachlässigt werden.

Verwenden Sie für SPE-ED die Webserver Facility und legen Sie ein neues Software Spec Template an. Ändern Sie auch die Service Unit und Service Time der einzelnen Devices. Für die CPU bietet sich z.B. als Einheit „Megainstruktionen“ an, für die Festplatte „MB from HD“ an.

Zeitstempel auf Extrablatt angeben: Ressourcenumgebung

Entwurfsalternativen

Modellieren Sie die Entwurfsalternativen erst, wenn das Ausgangsystem vollständig modelliert ist. Lassen Sie Ihr Ausgangssystem erst überprüfen, bevor Sie die Entwurfsalternativen modellieren.

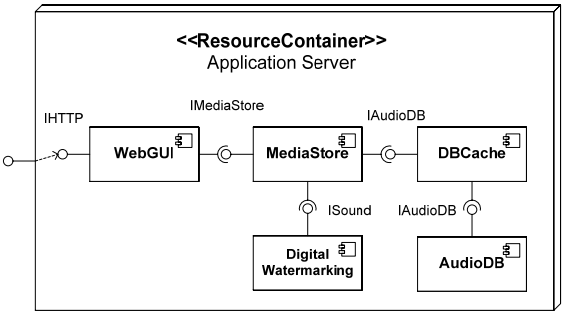
Entwurfsalternative 1: Cache

Motivation

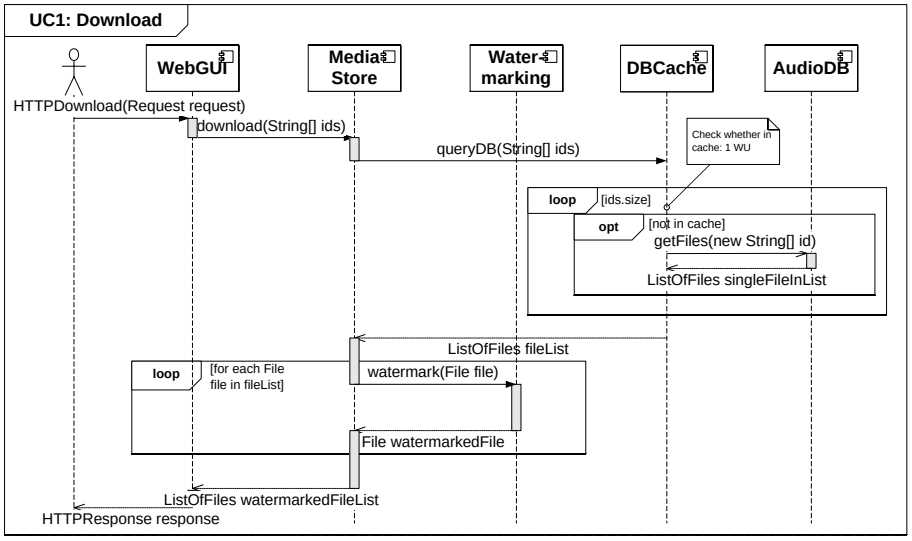
Es soll ein Cache eingesetzt werden, der einige Musikstücke zwischenspeichert und somit in manchen Fällen den Datenbankzugriff einsparen kann. Allerdings benötigt das Überprüfen des Caches selbst wieder ein wenig Rechenleistung.

Anpassung des Systems

Eine zusätzliche Komponente `DBCache` wird in das System eingeführt. Die folgenden Diagramme zeigen das angepasste System und den angepassten Kontrollfluss.



Ablauf mit zusätzlichem Resource Demand (nicht dargestellte Resource Demands bleiben unverändert):



Die Komponente `DBCache` erhält vom `MediaStore` die Liste aller angeforderten Musikstücke. Für jedes Musikstück wird geprüft, ob es im Cache vorliegt. Falls nicht, wird es einzeln von der Datenbank angefordert. `AudioDB.getFiles` wird also pro Schleifendurchlauf nur mit der einen jeweiligen id in der Liste aufgerufen und liefert eine Liste mit nur dem einen entsprechenden Musikstück zurück. Durch die Schleife wird dann die gesamte Liste zusammengestellt.

Cache Hit Ratio

Obwohl der Cache nur wenige Musikstücke im Verhältnis zur Gesamtgröße der Datenbank fassen kann, wird vermutet, dass ein Anteil der angefragten Musikstücke bereits im Cache vorgehalten sein wird. Die Strategie des Caches wird dazu auch Informationen zu den aktuellen Charts verwenden, da beliebte, aktuelle Musikstücke viel häufiger nachgefragt werden und somit im Cache vorgehalten werden können. Es wird davon ausgegangen, dass so eine Cache Hit Ratio von 20 % erreicht werden kann.

Rechenaufwand zum Prüfen des Caches

Das Prüfen, ob ein Musikstück im Cache vorliegt, kann dank eines Indexes mit nur 1 Work Units erledigt werden.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 1

Entwurfsalternative 2: Pool für Datenbankverbindungen

Motivation

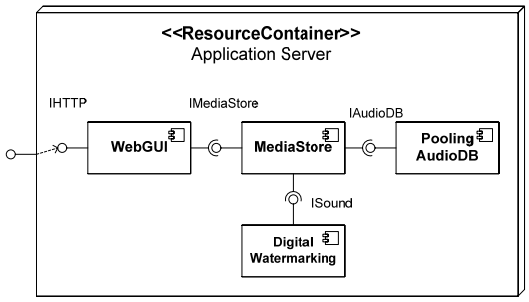
Um die Zeit für den Zugriff auf die interne Datenbank der `AudioDB` Komponente zu senken, kann ein Pool für Datenbankverbindungen eingeführt werden. Dabei muss dann nicht für jede Anfrage an die Datenbank eine neue Datenbankverbindung geöffnet werden, sondern es wird eine Datenbankverbindung aus dem Pool verwendet und nach der Benutzung wieder freigegeben. Allerdings kann so nur noch eine gewisse Anzahl von Threads gleichzeitig auf die Datenbank zugreifen.

Anpassung des Systems

In diesem Fall soll ein Pool der Größe 5 eingeführt werden. Die bestehende Komponente `AudioDB` wird ersetzt durch die Komponente `PoolingAudioDB`, die diese 5 Datenbankverbindungen verwaltet. Zusätzlich zu dem oben beschrieben Verhalten wird vor dem Zugriff auf die interne Datenbank der `AudioDB` Komponente eine Datenbankverbindung reserviert, und nach dem Datenbankzugriff wieder freigegeben.

Angaben zum Rechenaufwand

Dadurch verringert sich der Rechenaufwand für die Abfrage der Datenbank auf 1 Work Unit, für das Schreiben in die Datenbank (Use Case Upload) auf 5 Work Units.



Hinweise SPE-ED

In SPE-ED können solche Pools nicht direkt modelliert werden. Sie müssen bei dieser Alternative abschätzen, wie lange die Anfrage eines Benutzers durch die Verwendung des Pools verzögert wird. Dies hängt von der Ankunftsrate der Benutzer sowie der Dauer, wie lange die Ressource blockiert ist, ab.

Geben Sie auch hier an, wie Sie die Verzögerung abschätzen: _____ Millisekunden

Begründung:

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 2

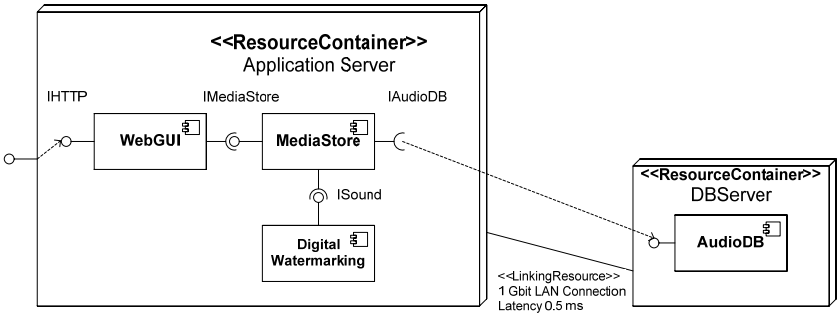
Entwurfsalternative 3: Zweiten Server für die Datenbankkomponenten

Motivation

Es wird ein zweiter Server bereitgestellt, auf den die AudioDB ausgelagert werden soll.

Anpassung des Systems

Die folgenden Diagramme zeigen das angepasste Deployment, der Kontrollfluss und das System verändern sich nicht.



Angaben zur Ressourcenumgebung

Der Datenbankserver hat dieselben Leistungsmerkmale wie der Applikationsserver.

Verwenden Sie für SPE-ED das vorgefertigte Facility Template DBServer und passen Sie die Overhead Matrix auch hier an.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 3

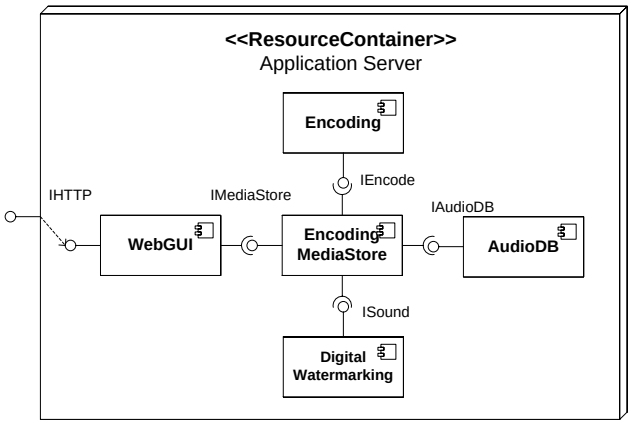
Entwurfsalternative 4: Bitrate senken

Motivation

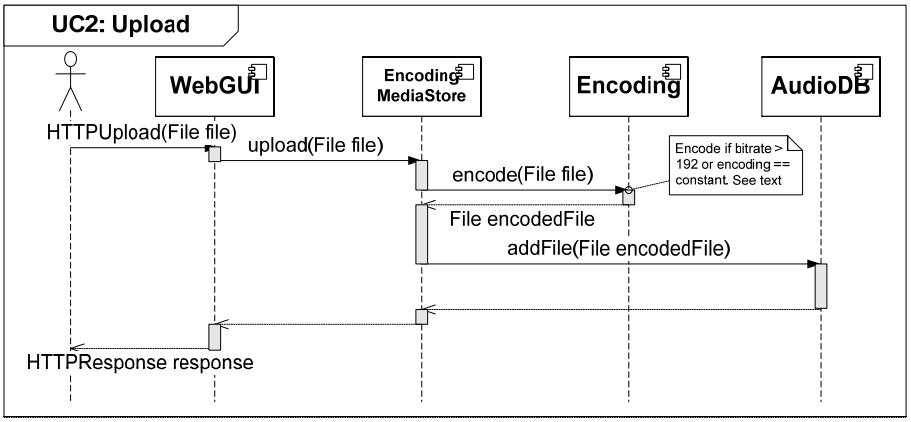
Musikstücke werden in unterschiedlicher Qualität hochgeladen. Weiterhin bietet das MP3 Format die Möglichkeit, Musikstücke mit konstanter oder variabler Bitrate zu kodieren, wobei bei der Verwendung einer variablen Bitrate in den meisten Fällen kleinere Dateien derselben Qualität erzeugt werden können. Um Speicherplatz und Zeit beim Herunterladen zu sparen, kann die Bitrate beim Hochladen verändert werden: Zum einen können für Musikstücke mit hohen durchschnittlichen Bitraten die durchschnittliche Bitrate gesenkt werden, wodurch allerdings die Qualität der Stücke sinkt. Weiterhin können Musikstücke, die mit einer konstanten Bitrate erzeugt wurden, unter Verwendung einer variablen Bitrate umgewandelt werden. Dabei sinkt die Qualität kaum.

Anpassung des Systems

Eine zusätzliche Komponente `Encoding` wird in die Architektur eingeführt. Die folgenden Diagramme zeigen das angepasste System und den angepassten Kontrollfluss für den Upload Anwendungsfall.



Ablauf mit zusätzlichem Resource Demand (nicht dargestellte Resource Demands bleiben unverändert):



Die Komponente `Encoding` komprimiert ein Musikstück nicht immer, sondern nur, wenn die Bitrate a) konstant ist oder b) höher als 192 kbps ist. Beide Fälle können zusammen auftreten, die Effekte werden kombiniert.

- Wenn die Bitrate konstant ist, kann durch die Umwandlung in eine mit variabler Bitrate kodierte MP3 Datei die Größe der Datei auf durchschnittlich 70% ihrer ursprünglichen Größe reduziert werden.
- Wenn die Bitrate höher als 192 kbps, wird sie auf 192 kbps gesenkt. Es kann davon ausgegangen werden, dass die Größe der Datei im selben Verhältnis abnimmt wie die Bitrate.

Wenn beide Fälle zusammen auftreten, wird trotzdem das Musikstück nur einmal neu kodiert, d.h. es fällt nur einmal der Rechenaufwand an. Die Größe des Musikstücks wird dann um 70% und um das Verhältnis der neuen zur alten Bitrate abnimmt.

Rechenaufwand zum Neukodieren (für Upload)

Wenn ein Musikstück neu kodiert werden soll, dekodiert die `Encoding` Komponente das Musikstück zunächst in das .wav Format, was einen Rechenaufwand von 10 Work Units pro MB der Ausgangsgröße bedeutet. Danach wird die Datei mit der neuen Bitrate kodiert, dies benötigt 10 Work Units pro MB der resultierenden MP3.

Falls die Bitrate des Musikstücks nicht geändert wird, wenn also die Bitrate variabel und nicht größer als 192 kbps ist, fällt kein Rechenaufwand an.

Veränderter Speicherbedarf (für Download)

Diese Rekodierung wirkt sich natürlich auch entsprechend auf die Dateigrößen der herunterzuladenen Musikstücke aus. Im Durchschnitt werden die Musikstücke also für die beiden gegebenen Benutzungsprofile um ca. 17% kleiner. Man erhält die folgende Verteilungsfunktion für die Dateigröße x in MB:

$$P(0{,}42 <= x < 0{,}84) = 0.051$$

$$P(0{,}84 <= x < 1{,}67) = 0.134$$

$$P(1{,}67 <= x < 2{,}5) = 0.193$$

$$P(2{,}5 <= x < 3{,}35) = 0.212$$

$$P(3{,}35 <= x < 4{,}2) = 0.224$$

$$P(4{,}2 <= x < 5) = 0.186$$

Der Mittelwert dieser Verteilung liegt bei 2,93 MB. Verwenden Sie die Verteilungsfunktion nur als Hilfe bei der Modellierung des Ladens von der Festplatte. Bei SPE-ED können Sie den Mittelwert auch für den Resource Demand beim Schreiben auf die Festplatte verwenden. Für die Modellierung des Rechenaufwands der Rekodierung können diese Angaben nicht verwendet werden, denn hier werden nicht alle Stücke neu kodiert. Es müssen für die Neukodierung also die Angaben der Aufzählung oben verwendet werden.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 4

Entwurfsalternative 5: Dynamischer Lookup

Motivation

Diese Entwurfsalternative hat im Gegensatz zu den bisherigen Alternativen nicht zum Ziel, die Performanz des Systems zu verbessern, sondern die Skalierbarkeit. Wenn alle Komponenten ihr Kommunikationspartner über einen dynamischen Lookup beim Broker finden, kann die Allokation der Komponenten leicht verändert werden, beispielsweise können neue Server angeschafft und einige der Komponenten darauf eingesetzt werden, ohne Änderungen am Code vornehmen zu müssen. Allerdings benötigt das dynamische Auffinden des Kommunikationspartners zusätzlichen Rechenaufwand.

Es wurde beschlossen, dass der dynamische Lookup nur verwendet werden soll, wenn sein Einsatz die Antwortzeit des Systems für die gegebenen Benutzungsprofile um nicht mehr als 10% verschlechtert. Analysieren Sie, ob dieses Performanzziel eingehalten werden kann und geben Sie an, ob diese Entwurfsalternative nach diesen Vorgaben eingesetzt werden kann.

Angaben zum Rechenaufwand

Der dynamische Lookup einer anderen Komponente kostet 17 Work Units.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 5

Fragestellungen

Untersuchen Sie das Ausgangssystem und die verschiedenen Entwurfsalternativen und sichern Sie die Ergebnisse wie oben unter Durchführung (Seite 2) angegeben.

1. Welche Entwurfsalternativen sollten eingesetzt werden*, welche nicht? Differenzieren Sie Ihre Antwort nach den beiden Benutzungsprofilen. Geben Sie Ihre Antwort und eine Begründung in der Tabelle 1 unten an.
   (*: Die Alternative 5 braucht die Antwortzeit nicht verbessern, darf sie aber nicht mehr als 10% verschlechtern, um eingesetzt zu werden.)
2. Angenommen, es wäre nicht möglich, alle sinnvollen Entwurfsalternativen umzusetzen, sondern nur einige, welche wäre das? Stellen Sie eine Reihenfolge der Entwurfsentscheidungen auf, um diese Frage zu beantworten, und sortieren Sie die Entwurfsentscheidungen in Tabelle 2, beginnend mit der nützlichsten. Beachten Sie dabei nicht eventuelle Abhängigkeiten der Entwurfsentscheidungen untereinander.

Entwurfsalternative	Benutzungsprofil	Durchschn. Antwortzeit	Einsetzen? Mit Begründung.
1: Cache	1		
	2		
2: Pool für Datenbankverbindungen	1		
	2		
3: Zweiter Server	1		
	2		
4: Bitrate senken	1		
	2		
5: Dynamischer Lookup	1		
	2		

Tabelle 1: Bewertung der Entwurfsalternativen

	Reihenfolge	
Entwurfsalternative	**Benutzungsprofil 1**	**Benutzungsprofil 2**
1: Cache		
2: Pool für Datenbankverbindungen		
3: Zweiter Server		
4: Bitrate senken		

Tabelle 2: Reihenfolge der Entwurfsalternativen

Kommentare zu den Entwurfsalternativen:

Zusatzfrage:

Bitte bearbeiten Sie diese Frage erst, wenn alle anderen Fragen beantwortet und die Antworten akzeptiert worden sind.

1. Wenn Sie beliebig viele Entwurfsalternativen umsetzen können, welche Kombination ergibt die besten Antwortzeiten?

Für diese Fragestellung müssen Sie verschiedene Kombinationen von Entwurfsalternativen zusammen analysieren und bewerten.

Beste Kombination:

B.1.2 Web Server

Palladio

Prof. Dr. Reussner, Software-Entwurf und -Qualität, Universität Karlsruhe (TH)
Praktikum Ingenieurmäßiger Software-Entwurf, SoSe 2007

Palladio

Name: _____

Fallstudie: Vergleich von Performanzvorhersageverfahren

Ein neuer komponentenbasierter Webserver soll entwickelt werden. Während der gesamten Entwicklung soll die Performanz des Entwurfs bewertet werden, um den späteren Benutzern ein System mit akzeptablen Antwortzeiten bieten zu können. Da Sie mit der Performanzanalyse vertraut sind, werden Sie gebeten, verschiedene Entwurfsalternativen hinsichtlich ihrer Performanz zu untersuchen.

Zeitstempel

Um untersuchen zu können, wie viel der Zeit auf welche Aufgabenteile entfällt, geben Sie bitte jeweils minutengenaue Zeiten für die Teilaufgaben an. Sie müssen dabei die Aufgabenteile nicht alle in der Reihenfolge abarbeiten, in der Sie sie hier vorfinden. Differenzieren Sie bei den Zeitstempeln weiterhin nach der Zeit für die reine Modellierung und der Zeit für die Fehlersuche.

Modellieren und analysieren Sie aber zunächst das Ausgangsystem vollständig und erst danach die Entwurfsalternativen, wenn Ihr Ausgangssystem abgenommen wurde. Führen Sie auch bei den Entwurfsalternativen erst die Analyse einer Entwurfsalternative durch, bevor Sie zur nächsten übergehen. Geben Sie Ihre Ergebnisse direkt in das Ergebnisblatt am Ende ein.

Checken Sie Ihre Lösung jeweils zu einem Zeitstempel in Ihren SVN Account ein, um ein Backup zu haben.

Zeitstempel auf Extrablatt angeben: Austeilung der Aufgabenstellung.

Übersicht über die Aufgabenstellung:

Durchführung der Untersuchungen

Modellieren und analysieren Sie zunächst das Ausgangsystem. Lassen Sie Ihre Modellierung und die Ergebnisse überprüfen, bevor Sie mit der Modellierung der Entwurfsalternativen beginnen.

Modellieren und analysieren Sie die Entwurfsalternativen in der hier angegebenen Reihenfolge. Analysieren Sie eine fertiggestellte Entwurfsalternative und tragen Sie die Ergebnisse in die Tabelle am Ende ein, bevor Sie mit der nächsten Entwurfsalternative fortfahren.

Arbeiten Sie in einem Verzeichnis Webserver-Nachname, das Sie bei jedem Zeitstempel (oder auch häufiger) in Ihr SVN Repository einchecken. Geben Sie in der Log Message den Namen des Zeitstempels an.

Palladio:

Untersuchen Sie mit dem PCM die Antwortzeit des Systems. Modellieren Sie hierfür die Parameter nur mit den für die Performanz relevanten Charakterisierungen. Nutzen Sie die vorhandenen Parameter der vorgegebenen Signaturen aus dem Repository für Charakterisierungen und verwenden Sie auch Komponentenparameter, wenn die Information nicht aus den Signaturenparametern geschlossen werden kann.

Wichtig: Laden Sie Modelle nur über „Browse Workspace“, da sie sonst absolut verlinkt sind und nicht auf anderen Systemen geöffnet werden können. Eine neue Eclipse Version ist benötigt, nicht nur neue Versionen der Plugins. Sie finden die Version im Wiki[1].

Legen Sie für die Entwurfsalternativen, bei denen andere Komponenten eingesetzt werden, ein neues System und eine neue Allokation an, die diese neue Komponente verwenden. Sie können dazu das alte System kopieren und anpassen. Die Allokation muss neu angelegt werden. Benennen Sie die neuen Modelle mit dem Suffix -EA-x, wobei x die Nummer der Alternative (0 für den Ausgangsentwurf) angibt. Sie erhalten beispielsweise das System Model File webserver-EA-1.system.

SEFFs müssen neu modelliert werden, sie können nicht aus anderen Komponenten kopiert werden.

Falls nur das Resource Environment und/oder die Allokation verändert wird, brauchen Sie nur diese Modelle neu zu erstellen. Auch hier kann das Resource Environment kopiert und angepasst werden. Verwenden Sie hier ebenfalls das oben angegebene Namensschema.

Legen Sie ein zweites Usage Model für das zweite Benutzungsprofil an. Für die Entwurfsalternative 2 benötigen Sie weiterhin zwei eigene Benutzungsprofile mit den angepassten Dateigrößen.

Erzeugen Sie weiterhin für jede Entwurfsalternative und jedes Benutzungsprofil eine eigene Run Config, die Sie ebenfalls nach dem obigen Schema benennen. Sie haben am Ende also 12 Run Configs angelegt, jeweils zwei pro Entwurfsalternative, nämlich pro Benutzungsprofil eines. Benennen Sie die Run Configs und die Simulationsergebnisse wie oben angegeben plus ein Suffix BP-y für das Benutzungsprofil y.

[1] http://sdqweb.ipd.uka.de/wiki/Praktikum_Ingenieurmäßiger_Software-Entwurf_SS07

Sie können die verschiedenen Analysen nicht parallel ausführen, sondern eine nach der anderen. Vergleichen Sie bei der Untersuchung des Benutzungsprofils 2 nur Ergebnisse, die mit der gleichen Simulationszeit erstellt wurden. Verändern Sie die Auflösung der Histogramme bzw. CDFs, um die Ergebnisse besser dargestellt zu sehen.

Speichern Sie die Analyseergebnisse für das Ausgangssystem und die einzelnen Entwurfsalternativen als Grafikdatei mit demselben Namensschema. Die Datenbank muss nicht gespeichert werden.

Bestehender Entwurf

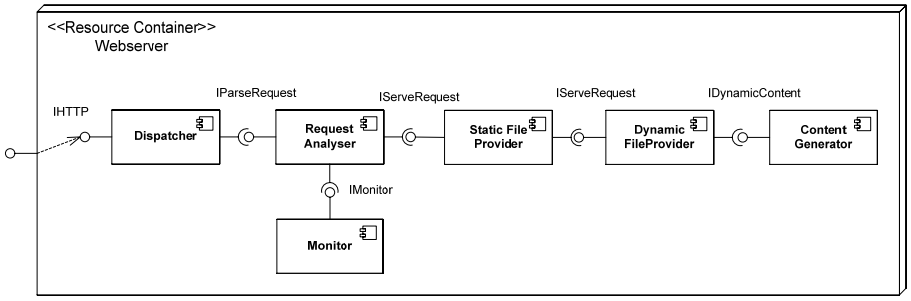


Abbildung 1: Ausgangssystem und Deployment

Das dazugehörige Repository mit den benötigten Komponenten, den Schnittstellen und ihren Signaturen wurden bereits im PCM modelliert. Laden Sie das Projekt aus dem Wiki[2] herunter, benennen Sie es nach dem Schema PCM_Webserver_Nachname um und vervollständigen Sie es mit den folgenden Angaben.

Interaktion zwischen Benutzer und Server:

Benutzer fordern eine Seite vom Webserver an. Dieser liefert zunächst die HTML Seite selbst zurück. Danach fragt der Browser des Benutzers die einzelnen Multimediainhalte, z.B. Bilder, die die HTML Seite referenziert, beim Webserver ab.

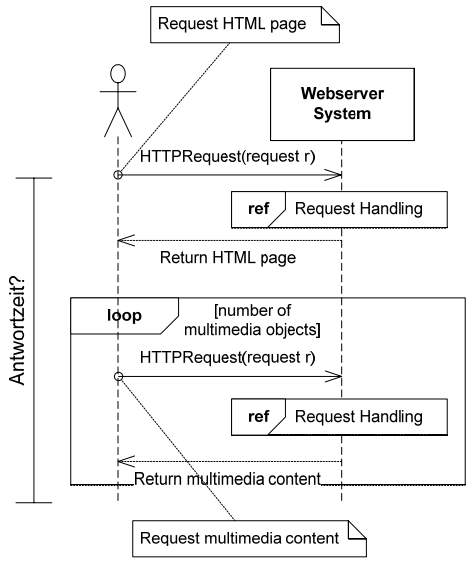


Abbildung 2: Sequenzdiagramm Interaktion Benutzer Server

Für diesen gesamten Ablauf soll die Antwortzeit ermittelt werden.

Interne Verarbeitung eines Requests

[2] http://sdqweb.ipd.uka.de/wiki/Praktikum_Ingenieurmäßiger_Software-Entwurf_SS07

In diesem Abschnitt wird zunächst der Ablauf innerhalb des Webserver Systems beschrieben. Die Resource Demands sind im Sequenzdiagramm „Request Handling“ (Abbildung 3) auf der nächsten Seite und im Abschnitt „Resource Demands“ auf Seite 7 aufgeführt.

Der Webserver selbst bietet nach außen nur eine Schnittstelle mit der Methode HTTPRequest an. Der gesendete HTTP Request wird zunächst von einer `Dispatcher` Komponente angenommen, die für diesen Request einen neuen Thread abspaltet, der die Bearbeitung übernimmt.

Im neuen Thread wird die `RequestAnalyser` Komponente aufgerufen. Der `RequestAnalyser` schickt zunächst eine Nachricht an den `Monitor`, der die Tatsache, dass ein Request eingetroffen ist und nun bearbeitet wird, notiert. Danach wird der Request zunächst geparst und dann zur Bearbeitung weitergegeben.

Die Komponenten `StaticFileProvider` und `DynamicFileProvider` sind in einer Zuständigkeitskette (Chain of Responsibility) angeordnet: Die Komponenten der Kette prüfen, ob sie für die Bearbeitung zuständig sind, wenn nicht, leiten sie die Anfrage weiter. Die Prüfung geschieht anhand der Dateiendung in der URL: Bei einer Dateiendung .html, .gif, .png usw. ist der `StaticFileProvider` zuständig. Bei der Dateiendung .php ist der `DynamicFileProvider` zuständig. Durch die Verwendung dieses Musters wird die Entscheidung, welche Komponente welche Anfragen bearbeitet, variabel gehalten.

Der Request wird also zunächst an den `StaticFileProvider` weitergegeben. Wenn es sich um ein statisch zu beantwortende Anfrage handelt, wie eine statische HTML Seite oder ein statischer Mutimediainhalt (Bild, Video, …), so wird die Anfrage direkt von dieser Komponente beantwortet. Andernfalls wird sie an den `DynamicFileProvider` weitergeleitet.

Der `DynamicFileProvider` beantwortet dynamische Anfragen. Dies können mit PHP generierte Seiten oder Multimediainhalte sein. Hierzu wird der Request analysiert und, je nach Inhalt, verschiedene Methoden des `ContentGenerator` aufgerufen. Wenn eine HTML Seite generiert werden soll, wird die Methode generateHTML(..) aufgerufen. Wenn ein Multimediainhalt generiert werden soll, wird die Methode generateMultimedia(..) aufgerufen. Der `ContentGenerator` generiert den HTML Code oder den angeforderten Multimediainhalt und liefert ihn zurück. Der Datentyp Content beinhaltet somit entweder ein HTML Dokument oder Multimediainhalte.

Andere Arten der Anfragen als statische und dynamische sind zur Zeit nicht vorgesehen und müssen auch nicht beachtet werden.

Der `RequestAnalyser` lässt nun vom Monitor einen weiteren Eintrag in die Logdatei vornehmen und erzeugt dann für den zurückgegebenen Content ein HTTP Response Objekt.

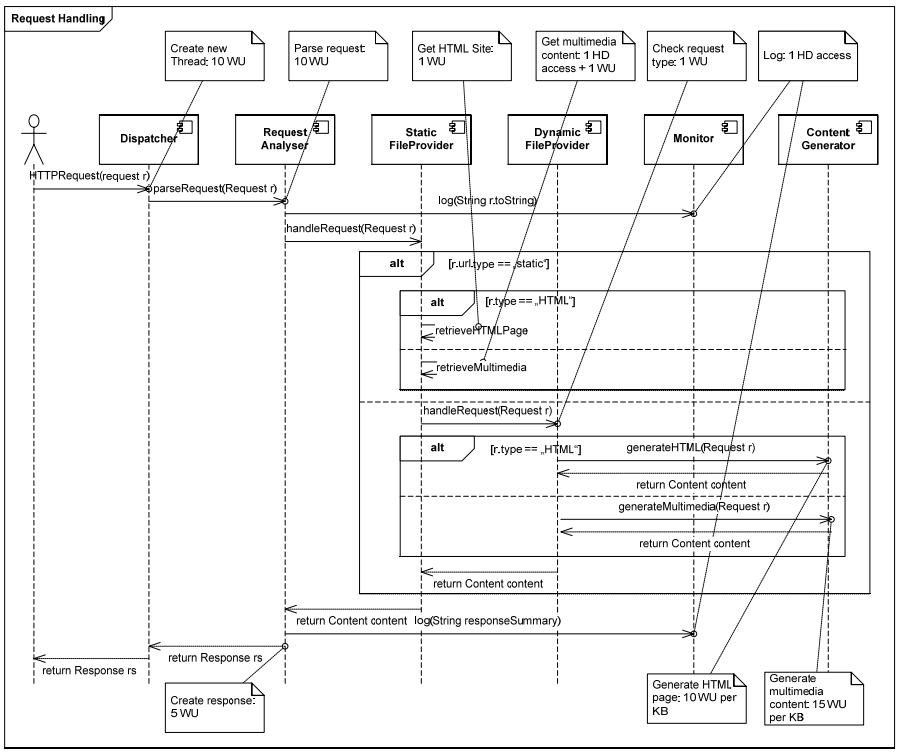


Abbildung 3: Sequenzdiagramm Request Handling

Die Übertragung von HTTP Request und Response über das Netzwerk wird vernachlässigt. Modellieren Sie so, als wäre dies in Nullzeit möglich. Insgesamt soll die Antwortzeit vom Absenden des Requests für die HTML Seite bis zum Erhalt des letzten Multimediaobjekts modelliert werden.

Zeitstempel auf Extrablatt angeben: Kontrollfluss

Resource Demands

Es folgen die Angaben zum Aufwand, die teilweise auch schon im Sequenzdiagramm oben enthalten sind:

`Dispatcher`

- Das Erzeugen eines neuen Threads benötigt 10 Work Units (WU). Da der Hauptthread keine weitere Aufgaben übernimmt, muss die Tatsache, dass der Kontrollfluss in diesem neuen Thread weitergeht, nicht explizit modelliert werden, nur der Aufwand für das Erzeugen des Threads ist von Interesse.

`RequestAnalyser`

- Das Parsen des HTTP Requests durch den `RequestAnalyser` benötigt 10 Work Units
- Das Erzeugen der HTTP Response aus dem gegebenen Content benötigt 5 Work Units.

`StaticFileProvider`

- Statische Seiten können schnell geliefert werden, da diese meistens bereits im Hauptspeicher des Webserver vorliegen. Sie müssen dann nur noch in die Datenstruktur Content überführt werden, was 1 Work Unit kostet.
- Das Liefern eines statischen Multimediaobjekts benötigt 1 Festplattenzugriff, weiterhin muss es ebenfalls in die Datenstruktur Content überführt werden, was 1 Work Unit kostet.

`DynamicFileProvider`

- Für dynamische Inhalte wird festgestellt, welche Art von Request vorliegt (HTML Seite oder Multimediainhalt, Kosten 1 Work Unit) und der Request dann an die entsprechende Methode der `ContentGenerator` Komponente weitergeleitet.

`ContentGenerator`

- Bei dynamischen HTML Seiten liegt der Aufwand zur Interpretierung bei 10 Work Units pro KB der resultierenden HTML Seite.
- Dynamische Multimediaobjekte benötigen 15 Work Units pro KB des resultierenden Multimediainhalts.

`Monitor`

- Das Erzeugen eines Eintrags in der Logdatei benötigt einen Festplattenzugriff.

Zeitstempel auf Extrablatt angeben: Resource Demands

Benutzungsprofile

Domänenexperten wurden zu Rate gezogen, um mögliche Benutzungsprofile zu ermitteln. Für einen Webserver ist dies natürlich nicht leicht zu bestimmen, da viele verschiedene Benutzungsprofile denkbar sind. Die Domänenexperten haben aber trotzdem zwei Benutzungsprofile definiert, die für die Bewertung der Performanz verwendet werden können:

Benutzungsprofil 1

Das erste Benutzungsprofil spiegelt eine Verwendung des Webservers mit viel Multimedia-Inhalt auf häufig statischen Seiten wieder.

- Es greift nur ein Benutzer zur Zeit auf das System zu.
- Closed Workload, Think Time 0
- In 40% der Fälle werden statische HTML Seiten abgefragt, in 60% der Fälle dynamische.
- Multimediainhalte sind zu 70% statisch und zu 30% dynamisch generiert.
- Anzahl der Multimediaobjekte pro Seite x:

$$P(x = 0) = 0{,}1$$

$$P(x = 1) = 0{,}1$$

$$P(x = 2) = 0{,}2$$

$$P(x = 3) = 0{,}3$$

$$P(x = 4) = 0{,}2$$

$$P(x = 5) = 0{,}1$$

- Bei den dynamischen Inhalten spielt die Größe der resultierenden Datei eine Rolle.

  Größe der resultierenden, dynamisch erzeugten HTML Seiten x in KB = 10^3 Byte:

$$P(5\ KB <= x < 10\ KB) = 0{,}3$$

$$P(10\ KB <= x < 15\ KB) = 0{,}5$$

$$P(15\ KB <= x < 20\ KB) = 0{,}2$$

  Größe der resultierenden, dynamisch erzeugten Multimediainhalte x in KB:

$$P(5\ KB <= x < 25\ KB) = 0{,}2$$

$$P(25\ KB <= x < 75\ KB) = 0{,}6$$

$$P(75\ KB <= x < 100\ KB) = 0{,}2$$

  Palladio: Modellieren Sie diese beiden Verteilungen als Komponentenparameter, da sie nicht aus dem Request geschlossen werden können.

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 1

Benutzungsprofil 2

Im zweiten Benutzungsprofil werden mehr dynamische textuelle Inhalte abgerufen, wie beispielsweise in einem Wiki. Weiterhin werden mehr Benutzer erwartet.

- Die Zwischenankunftszeit ist exponentialverteilt mit dem Erwartungswert 1 Sekunde. Tragen Sie in den StoEx des Open Workload einfach „Exp(1)“ ein.
- In 20% der Fälle werden statische HTML Seiten abgefragt, in 80% der Fälle dynamische.
- Es werden weniger Multimediainhalte pro Seite abgefragt. Anzahl der Multimediaobjekte pro Seite x:

$$P(x = 0) = 0{,}3$$

$$P(x = 1) = 0{,}3$$

$$P(x = 2) = 0{,}2$$

$$P(x = 3) = 0{,}1$$

$$P(x = 4) = 0{,}1$$

- Es werden mehr dynamische Multimediainhalte abgefragt. Multimediainhalte sind nun zu 60% statisch und zu 40% dynamisch generiert.

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 2

Ressourcenumgebung

Alle Komponenten sind auf einem Server eingesetzt, wie Abbildung 1 bereits zeigt. Die CPU des Systems ist ein AMD Athlon XP mit einer Taktrate von 1,145 GHz. Ein CPU Zyklus entspricht einer Instruktion. Für die Angaben wurde der Rechenaufwand in Work Units angegeben. Ein Work Unit entspricht ca. $1{,}145 * 10^6$ Instruktionen = $1{,}145 * 10^6$ CPU Zyklen. Es wurde gemessen, dass die Latenzzeit der Festplatte (Seek Time) 3,5 ms ist und sie weiterhin ca. 24 MB/s = $24 * 10^6$ Byte/s lesen bzw. schreiben kann.

Das Resource Environment ist für Palladio bereits vorgegeben. Sie müssen allerdings noch die Verarbeitungsgeschwindigkeiten der Geräte angeben. Für die CPU bietet sich z.B. als Einheit „Work Units pro Sekunde“ an. Für die Festplatte modellieren Sie nur die Latenz (Seek Time), der Durchsatz kann vernachlässigt werden. Beide Geräte haben bereits die benötigte Scheduling Policy „Processor Sharing“ voreingestellt.

Zeitstempel auf Extrablatt angeben: Ressourcenumgebung

Entwurfsalternativen

Modellieren Sie die Entwurfsalternativen erst, wenn das Ausgangsystem vollständig modelliert ist. Lassen Sie Ihr Ausgangssystem erst überprüfen, bevor Sie die Entwurfsalternativen modellieren.

Entwurfsalternative 1: Cache für dynamische Inhalte

Motivation

Die Erzeugung von Seiten mit dynamischem Inhalt erfordert Rechenzeit und muss im Normalbetrieb für jeden Request neu geschehen. Bei der Erzeugung von PHP Seiten muss beispielsweise zunächst das PHP Skript interpretiert werden und danach mit dem erzeugten ausführbaren PHP Code die tatsächliche HTML Seite generiert werden. Wenn jedoch dieselben Inhalte immer wieder angefragt werden, die außerdem rechenintensiv in der Erzeugung sind, kann es sinnvoll sein, einen Cache für dynamische Seiten einzusetzen. Ein Beispiel ist eAccelerator, einem sog. PHP Accelerator, der den ausführbaren PHP Code zwischenspeichert.

Anpassung des Systems

Die Komponente ContentGenerator wird in dieser Entwurfsalternative durch die Komponente AcceleratedContentGenerator ersetzt. Innerhalb dieser Komponente ist der Resource Demand zum Erzeugen eines dynamischen Inhalts nun anders als oben unter Ressourcenumgebung angegeben:

- 30 % des dynamischen Inhalts (Seiten und Multimediainhalte) liegen im Cache der AcceleratedContentGenerator Komponente bereits ausführbar vor. Für diese fallen nur 3 Work Units pro KB der resultierenden Datei für die Ausführung an.
- Für die anderen 70 % des dynamischen Inhalts wird weiterhin zuerst die Skriptdatei von der AcceleratedContentGenerator Komponente interpretiert und dann die Inhalte (HTML oder Multimediaobjekt) generiert. Der Aufwand zur Erzeugung liegt hier weiterhin bei 10 bzw. 15 Work Units pro KB der resultierenden Datei.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 1

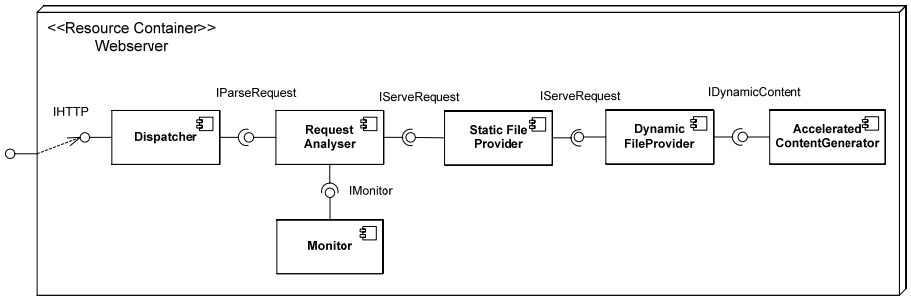


Abbildung 4: Entwurfsalternative 1: System und Deployment

Entwurfsalternative 2: Dynamischer Lookup

Motivation

Diese Entwurfsalternative hat im Gegensatz zu den bisherigen Alternativen nicht zum Ziel, die Performanz des Systems zu verbessern, sondern die Skalierbarkeit. Wenn alle Komponenten ihr Kommunikationspartner über einen dynamischen Lookup finden, kann die Allokation der Komponenten leicht verändert werden, beispielsweise können neue Server angeschafft und einige der Komponenten darauf eingesetzt werden, ohne Änderungen am Code vornehmen zu müssen. Allerdings benötigt das dynamische Auffinden des Kommunikationspartners zusätzlichen Rechenaufwand.

Es wurde beschlossen, dass der dynamische Lookup nur verwendet werden soll, wenn sein Einsatz die Antwortzeit des Systems für die gegebenen Benutzungsprofile um nicht mehr als 10% verschlechtert. Analysieren Sie, ob dieses Performanzziel eingehalten werden kann und geben Sie an, ob diese Entwurfsalternative nach diesen Vorgaben eingesetzt werden kann.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 2

Entwurfsalternative 3: Logging parallelisieren

Motivation

Eine weitere Alternative zur Reduzierung des Aufwands, das Logging über die Monitor-Komponente wenn möglich parallel zum eigentlichen Kontrollfluss auszuführen.

Anpassung des Systems

Dafür wird eine neue Komponente FastRequestAnalyser eingeführt, die die RequestAnalyser Komponente ersetzt und sich wie unten angeben verhält. Der erste Aufruf der Monitor-Komponente kann parallel zur Weiterverarbeitung des Requests durch den Static- bzw. DynamicFileProvider geschehen. Für den zweiten Aufruf werden die Ergebnisse des StaticFileProviders für die Zusammenfassung der Response benötigt. Der zweite Aufruf kann also nur parallel zur Generierung der Response durchgeführt werden. Um dies zu verdeutlichen, wurde im unten angegebenen Sequenzdiagramm die Generierung der HTTP Response diesmal explizit modelliert.

Um parallel arbeiten zu können, muss ein Thread für das Logging vom Haupt-Thread abgespalten werden, der Resource Demand dafür ist im Abschnitt Resource Demands auf Seite 7 angegeben. Dieser Thread kann für den zweiten parallelen Abschnitt wiederverwendet werden, der Aufwand für das Erzeugen eines neuen Threads fällt also nur einmal an.

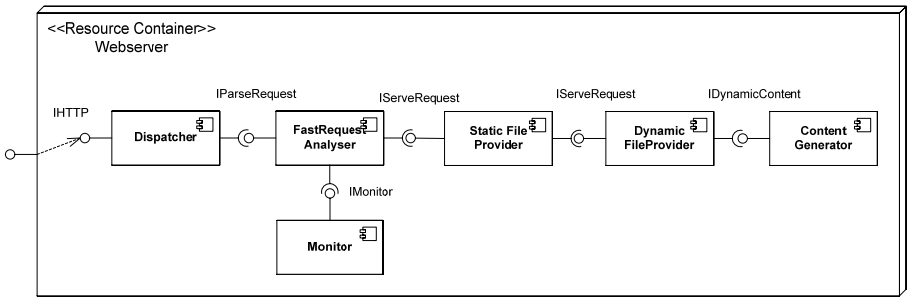


Abbildung 5: Entwurfsalternative 3: System und Deployment

Sequenzdiagramm auf der nächsten Seite.

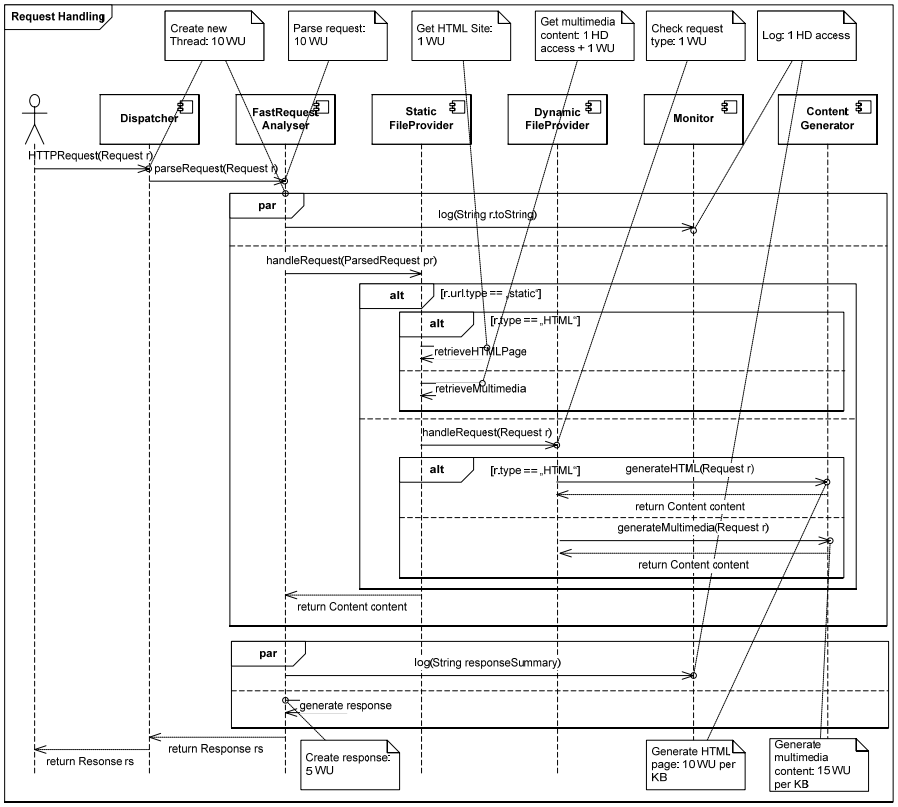


Abbildung 6: Entwurfsalternative 3: Veränderter Ablauf

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 3

Entwurfsalternative 4: Hardware Replikation mit einem zweiten Server

Motivation

Ein Teil der Komponenten wird in dieser Entwurfsalternative auf einen zweiten Server ausgelagert. In diesem Fall wurde beschlossen, die Komponenten DynamicFileProvider und ContentGenerator auf den zweiten Server auszulagern, so dass der erste Server die Anfragen annimmt, analysiert und statische Anfragen bedient, und der zweite dynamische Anfragen.

Angaben zur Ressourcenumgebung

Der zweite Server (AppServer) hat eine doppelt so schnelle CPU wie der erste. Die Festplatte hat die gleichen Leistungsmerkmale.

Das folgende Diagramm zeigt die neue Allokation des Systems:

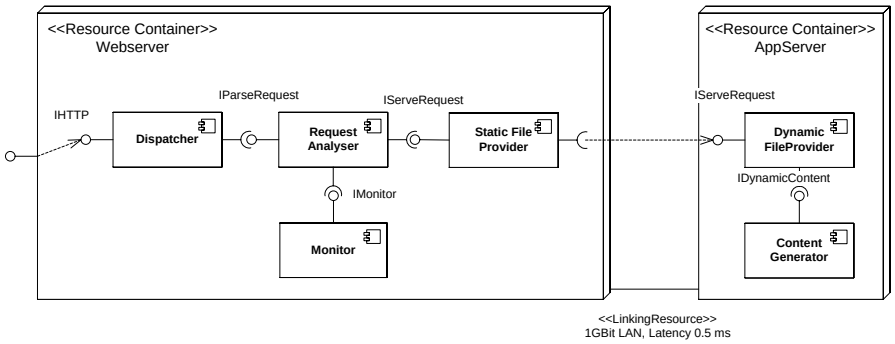


Abbildung 7: Entwurfsalternative 4: System und Deployment

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 4

Entwurfsalternative 5: Thread Pool

Motivation

Im Normalbetrieb erstellt der Webserver in der Dispatcher Komponente für jede Anfrage einen neuen Thread. Die Erzeugung eines Threads ist dabei mit Rechenaufwand verbunden.

Eine Entwurfsalternative ist die Einführung eines Thread Pools. Hier hält der Webserver bereits eine Anzahl Threads vor, und weist jedem Request dann nur noch einen Thread zu. Ein Vorteil dieser Entwurfsalternative ist, dass bei einer hohen Last nicht mehr zu viele Threads gleichzeitig aktiv sind und durch ständigen Kontextwechsel die Ressourcen nur suboptimal ausnutzen. Weiterhin wird die Zeit für die Erzeugung eines neuen Threads eingespart.

Anpassung des Systems

Modellieren Sie einen Threadpool der Größe 8. Gehen Sie dabei davon aus, dass mit dem Holen eines Threads aus dem Threadpool kein Rechenaufwand verbunden ist.

Diese Entwurfsalternative wird durch eine neue Komponente PoolingDispatcher realisiert, die die Komponente Dispatcher ersetzt.

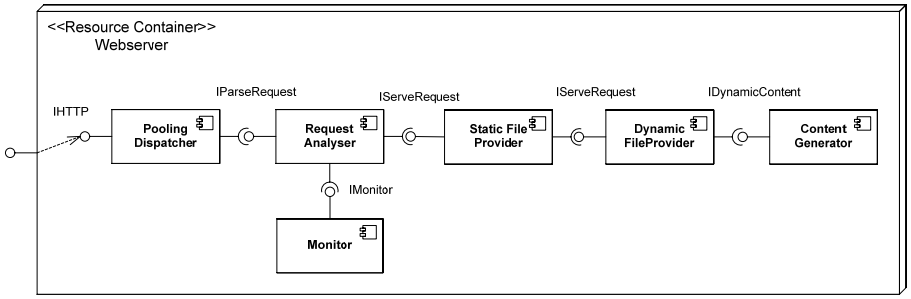


Abbildung 8: Entwurfsalternative 5: System und Deployment

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 5

Fragestellungen

Untersuchen Sie das Ausgangssystem und die verschiedenen Entwurfsalternativen und sichern Sie die Ergebnisse wie oben unter Durchführung (Seite 2) angegeben.

1. Welche Entwurfsalternativen sollten eingesetzt werden*, welche nicht? Differenzieren Sie Ihre Antwort nach den beiden Benutzungsprofilen. Geben Sie Ihre Antwort und eine Begründung in der Tabelle 1 unten an.
   (*: Die Alternative 2 braucht die Antwortzeit nicht verbessern, darf sie aber nicht mehr als 10% verschlechtern, um eingesetzt zu werden.)
2. Angenommen, es wäre nicht möglich, alle sinnvollen Entwurfsalternativen umzusetzen, sondern nur einige, welche wäre das? Stellen Sie eine Reihenfolge der Entwurfsentscheidungen auf, um diese Frage zu beantworten, und sortieren Sie die Entwurfsentscheidungen in Tabelle 2, beginnend mit der nützlichsten. Beachten Sie dabei nicht eventuelle Abhängigkeiten der Entwurfsentscheidungen untereinander.

Verschiedene Zeitstempel auf Extrablatt angeben: Analyse

Entwurfsalternative	Benutz-ungs-profil	Einsetzen? Mit Begründung.
1: Cache für dyn. Inhalte	1	
	2	
2: Dynamischer Lookup	1	
	2	
3: Logging parallelisieren	1	
	2	
4: Hardware Replikation	1	
	2	
5: Thread Pool	1	
	2	

Tabelle 1: Bewertung der Entwurfsalternativen

	Reihenfolge	
Entwurfsalternative	**Benutzungsprofil 1**	**Benutzungsprofil 2**
1: Cache für dyn. Inhalte		
3: Logging parallelisieren		
4: Hardware Replikation		
5: Thread Pool		

Tabelle 2: Reihenfolge der Entwurfsalternativen

Kommentare zu den Entwurfsalternativen:

SPE

Prof. Dr. Reussner, Software-Entwurf und -Qualität, Universität Karlsruhe (TH)
Praktikum Ingenieurmäßiger Software-Entwurf, SoSe 2007

SPE-ED

Name: _____

Fallstudie: Vergleich von Performanzvorhersageverfahren

Ein neuer komponentenbasierter Webserver soll entwickelt werden. Während der gesamten Entwicklung soll die Performanz des Entwurfs bewertet werden, um den späteren Benutzern ein System mit akzeptablen Antwortzeiten bieten zu können. Da Sie mit der Performanzanalyse vertraut sind, werden Sie gebeten, verschiedene Entwurfsalternativen hinsichtlich ihrer Performanz zu untersuchen.

Zeitstempel

Um untersuchen zu können, wie viel der Zeit auf welche Aufgabenteile entfällt, geben Sie bitte jeweils minutengenaue Zeiten für die Teilaufgaben an. Sie müssen dabei die Aufgabenteile nicht alle in der Reihenfolge abarbeiten, in der Sie sie hier vorfinden. Differenzieren Sie bei den Zeitstempeln weiterhin nach der Zeit für die reine Modellierung und der Zeit für die Fehlersuche.

Modellieren und analysieren Sie aber zunächst das Ausgangsystem vollständig und erst danach die Entwurfsalternativen, wenn Ihr Ausgangssystem abgenommen wurde. Führen Sie auch bei den Entwurfsalternativen erst die Analyse einer Entwurfsalternative durch, bevor Sie zur nächsten übergehen. Geben Sie Ihre Ergebnisse direkt in das Ergebnisblatt am Ende ein.

Checken Sie Ihre Lösung jeweils zu einem Zeitstempel in Ihren SVN Account ein, um ein Backup zu haben.

Zeitstempel auf Extrablatt angeben: Austeilung der Aufgabenstellung.

Übersicht über die Aufgabenstellung:

Durchführung der Untersuchungen

Modellieren und analysieren Sie zunächst das Ausgangsystem. Lassen Sie Ihre Modellierung und die Ergebnisse überprüfen, bevor Sie mit der Modellierung der Entwurfsalternativen beginnen.

Modellieren und analysieren Sie die Entwurfsalternativen in der hier angegebenen Reihenfolge. Analysieren Sie eine fertiggestellte Entwurfsalternative und tragen Sie die Ergebnisse in die Tabelle am Ende ein, bevor Sie mit der nächsten Entwurfsalternative fortfahren.

Arbeiten Sie in einem Verzeichnis Webserver-Nachname, das Sie bei jedem Zeitstempel (oder auch häufiger) in Ihr SVN Repository einchecken. Geben Sie in der Log Message den Namen des Zeitstempels an.

SPE

Untersuchen Sie mit SPE-ED die durchschnittliche Antwortzeit.

Legen Sie für jede Entwurfsalternative (EA) und Benutzungsprofil ein eigenes Projekt an. Sie können die Projekte auch in unterschiedlichen Verzeichnissen in eigenen SPE-ED Installationen ablegen, damit Sie das Ausgangsprojekt kopieren und weiterverwenden können. Nennen Sie entweder die Projekte Webserver-EA-x-BP-y, wobei x die Nummer der Alternative (0 für den Ausgangsentwurf) und y die Nummer des Benutzungsprofils angibt, oder benennen Sie die Verzeichnisse nach diesem Schema.

Wichtig: Bedenken Sie, dass Sie nur vier Navigationsboxen zur Verfügung haben und Sie nicht zu viele Expand Nodes einsetzen können. Gehen Sie also sparsam mit Expand Nodes um, indem Sie Sequenzen innerhalb einer Expand Node zusammenfassen und als eine Basic Node modellieren. Nur für Schleifen und Verzweigungen sollten Expand Nodes verwendet werden, aber auch hier nicht immer. Überlegen Sie, wie Sie die verfügbaren Navigationsboxen verwenden, bevor Sie mit der Modellierung beginnen.

Bewerten Sie die Entwurfsalternativen aufgrund der Antwortzeiten für beide Anwendungsfälle. Der einfachste Weg hierfür ist es, jeweils beide Anwendungsfälle in einem Szenario zu modellieren. Sie haben dann den Vorteil, dass Sie nur die „Scenario Contention Analyse“ durchführen müssen. Für Entwurfsalternative 1 benötigen Sie ein weiteres Szenario auf einer weiteren Facility. Untersuchen Sie die Antwortzeit für dieses Szenario und setzen Sie das Ergebnis in einen Synchronisationsknoten im Ausgangsszenario als Delay ein. Auch hier benötigen Sie also keine Simulation.

Prüfen Sie bei jeder Analyse, ob immer noch die richtige Analyseart ausgewählt ist.

Speichern Sie einen Screenshot der Analyseergebnisse für das Ausgangssystem und die einzelnen Entwurfsalternativen als Grafikdatei mit demselben Namensschema. Insgesamt müssen Sie 12 Dateien speichern, durch die Kombination von 6 Entwurfsalternativen (inklusive dem Ausgangssystem als EA0) mit zwei Benutzungsprofilen.

Bestehender Entwurf

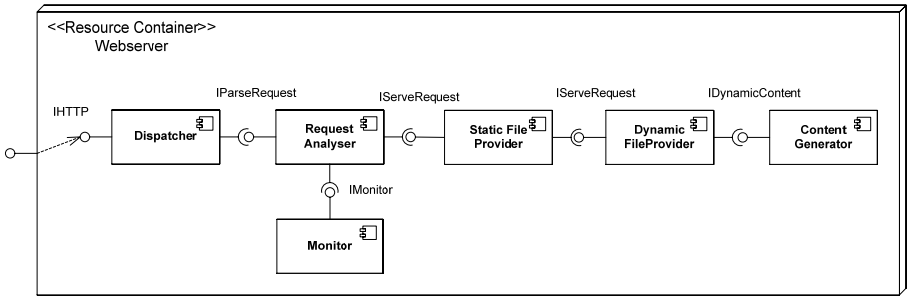


Abbildung 1: Ausgangssystem und Deployment

Interaktion zwischen Benutzer und Server:

Benutzer fordern eine Seite vom Webserver an. Dieser liefert zunächst die HTML Seite selbst zurück. Danach fragt der Browser des Benutzers die einzelnen Multimediainhalte, z.B. Bilder, die die HTML Seite referenziert, beim Webserver ab.

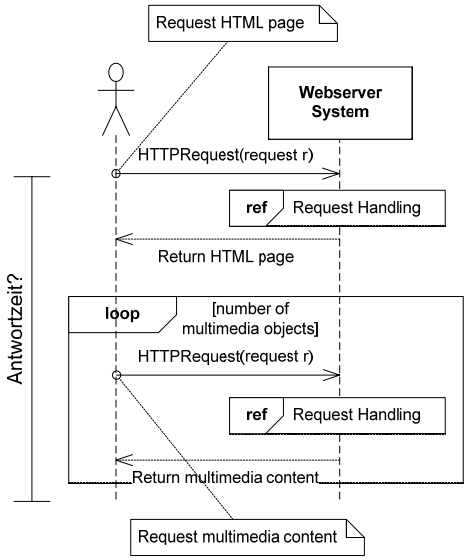


Abbildung 2: Sequenzdiagramm Interaktion Benutzer Server

Für diesen gesamten Ablauf soll die Antwortzeit ermittelt werden.

Interne Verarbeitung eines Requests

In diesem Abschnitt wird zunächst der Ablauf innerhalb des Webserver Systems beschrieben. Die Resource Demands sind im Sequenzdiagramm „Request Handling“ (Abbildung 3) auf der nächsten Seite und im Abschnitt „Resource Demands“ auf Seite 6 aufgeführt.

Der Webserver selbst bietet nach außen nur eine Schnittstelle mit der Methode HTTPRequest an. Der gesendete HTTP Request wird zunächst von einer `Dispatcher` Komponente angenommen, die für diesen Request einen neuen Thread abspaltet, der die Bearbeitung übernimmt.

Im neuen Thread wird die `RequestAnalyser` Komponente aufgerufen. Der `RequestAnalyser` schickt zunächst eine Nachricht an den `Monitor`, der die Tatsache, dass ein Request eingetroffen ist und nun bearbeitet wird, notiert. Danach wird der Request zunächst geparst und dann zur Bearbeitung weitergegeben.

Die Komponenten `StaticFileProvider` und `DynamicFileProvider` sind in einer Zuständigkeitskette (Chain of Responsibility) angeordnet: Die Komponenten der Kette prüfen, ob sie für die Bearbeitung zuständig sind, wenn nicht, leiten sie die Anfrage weiter. Die Prüfung geschieht anhand der Dateiendung in der URL: Bei einer Dateiendung .html, .gif, .png usw. ist der `StaticFileProvider` zuständig. Bei der Dateiendung .php ist der `DynamicFileProvider` zuständig. Durch die Verwendung dieses Musters wird die Entscheidung, welche Komponente welche Anfragen bearbeitet, variabel gehalten.

Der Request wird also zunächst an den `StaticFileProvider` weitergegeben. Wenn es sich um ein statisch zu beantwortende Anfrage handelt, wie eine statische HTML Seite oder ein statischer Mutimediainhalt (Bild, Video, …), so wird die Anfrage direkt von dieser Komponente beantwortet. Andernfalls wird sie an den `DynamicFileProvider` weitergeleitet.

Der `DynamicFileProvider` beantwortet dynamische Anfragen. Dies können mit PHP generierte Seiten oder Multimediainhalte sein. Hierzu wird der Request analysiert und, je nach Inhalt, verschiedene Methoden des `ContentGenerator` aufgerufen. Wenn eine HTML Seite generiert werden soll, wird die Methode generateHTML(..) aufgerufen. Wenn ein Multimediainhalt generiert werden soll, wird die Methode generateMultimedia(..) aufgerufen. Der `ContentGenerator` generiert den HTML Code oder den angeforderten Multimediainhalt und liefert ihn zurück. Der Datentyp Content beinhaltet somit entweder ein HTML Dokument oder Multimediainhalte.

Andere Arten der Anfragen als statische und dynamische sind zur Zeit nicht vorgesehen und müssen auch nicht beachtet werden.

Der `RequestAnalyser` lässt nun vom Monitor einen weiteren Eintrag in die Logdatei vornehmen und erzeugt dann für den zurückgegebenen Content ein HTTP Response Objekt.

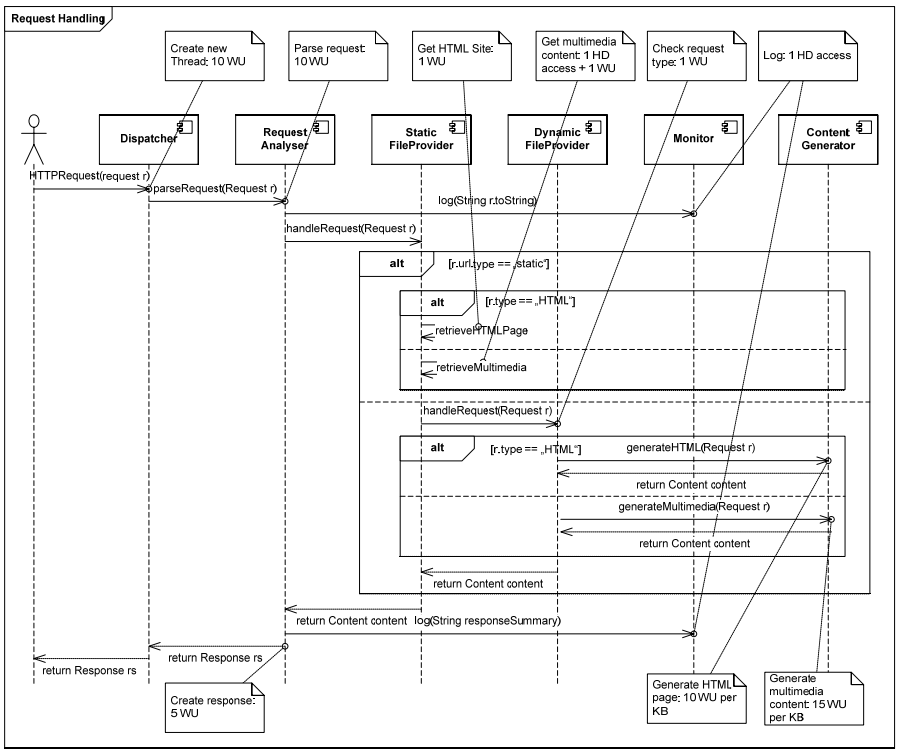


Abbildung 3: Sequenzdiagramm Request Handling

Die Übertragung von HTTP Request und Response über das Netzwerk wird vernachlässigt. Modellieren Sie so, als wäre dies in Nullzeit möglich. Insgesamt soll die Antwortzeit vom Absenden des Requests für die HTML Seite bis zum Erhalt des letzten Multimediaobjekts modelliert werden.

Zeitstempel auf Extrablatt angeben: Kontrollfluss

Resource Demands

Es folgen die Angaben zum Aufwand, die teilweise auch schon im Sequenzdiagramm oben enthalten sind:

`Dispatcher`

- Das Erzeugen eines neuen Threads benötigt 10 Work Units (WU). Da der Hauptthread keine weitere Aufgaben übernimmt, muss die Tatsache, dass der Kontrollfluss in diesem neuen Thread weitergeht, nicht explizit modelliert werden, nur der Aufwand für das Erzeugen des Threads ist von Interesse.

`RequestAnalyser`

- Das Parsen des HTTP Requests durch den `RequestAnalyser` benötigt 10 Work Units
- Das Erzeugen der HTTP Response aus dem gegebenen Content benötigt 5 Work Units.

`StaticFileProvider`

- Statische Seiten können schnell geliefert werden, da diese meistens bereits im Hauptspeicher des Webserver vorliegen. Sie müssen dann nur noch in die Datenstruktur Content überführt werden, was 1 Work Unit kostet.
  Das statische Multimediaobjekt wird von der Festplatte geladen, d.h. es wird 1 Festplattenzugriff benötigt. Der Mittelwert der Größe der geladenen Multimediaobjekte ist 50 KB. Weiterhin muss es ebenfalls in die Datenstruktur Content überführt werden, was 1 Work Unit kostet.

`DynamicFileProvider`

- Für dynamische Inhalte wird festgestellt, welche Art von Request vorliegt (HTML Seite oder Multimediainhalt, Kosten 1 Work Unit) und der Request dann an die entsprechende Methode der `ContentGenerator` Komponente weitergeleitet.

`ContentGenerator`

- Bei dynamischen HTML Seiten liegt der Aufwand zur Interpretierung bei 10 Work Units pro KB der resultierenden HTML Seite.
- Dynamische Multimediaobjekte benötigen 15 Work Units pro KB des resultierenden Multimediainhalts.

`Monitor`

- Das Erzeugen eines Eintrags in der Logdatei benötigt einen Festplattenzugriff. Es werden ca. 240 Byte pro Zugriff geschrieben.

Zeitstempel auf Extrablatt angeben: Resource Demands

Benutzungsprofile

Domänenexperten wurden zu Rate gezogen, um mögliche Benutzungsprofile zu ermitteln. Für einen Webserver ist dies natürlich nicht leicht zu bestimmen, da viele verschiedene Benutzungsprofile denkbar sind. Die Domänenexperten haben aber trotzdem zwei Benutzungsprofile definiert, die für die Bewertung der Performanz verwendet werden können:

Benutzungsprofil 1

Das erste Benutzungsprofil spiegelt eine Verwendung des Webservers mit viel Multimedia-Inhalt auf häufig statischen Seiten wieder.

- Es greift nur ein Benutzer zur Zeit auf das System zu.
- No Contention Lösung oder Closed Workload, Think Time 0
- In 40% der Fälle werden statische HTML Seiten abgefragt, in 60% der Fälle dynamische.
- Multimediainhalte sind zu 70% statisch und zu 30% dynamisch generiert.
- Anzahl der Multimediaobjekte pro Seite: Mittelwert 2,5
- Bei den dynamischen Inhalten spielt die Größe der resultierenden Datei eine Rolle.
  Größe der resultierenden, dynamisch erzeugten HTML Seiten x in KB = 10^3 Byte:
  - Mittelwert 12 KB

  Größe der resultierenden, dynamisch erzeugten Multimediainhalte x in KB:
  - Mittelwert 50 KB

- *Die Verteilungen werden nicht direkt benötigt, nur ggf. für die Abschätzung in Entwurfsalternative 5: Thread Pool:*

  Verteilung der Größe der resultierenden, dynamisch erzeugten HTML Seiten

  $$P(5\ KB <= x < 10\ KB) = 0{,}3$$
  $$P(10\ KB <= x < 15\ KB) = 0{,}5$$
  $$P(15\ KB <= x < 20\ KB) = 0{,}2$$

  Verteilung der Größe der resultierenden, dynamisch erzeugten Multimediainhalte

  $$P(5\ KB <= x < 25\ KB) = 0{,}2$$
  $$P(25\ KB <= x < 75\ KB) = 0{,}6$$
  $$P(75\ KB <= x < 100\ KB) = 0{,}2$$

  Verteilung der Anzahl von Multimediaobjekten x pro Seite:

  $$P(x = 0) = 0{,}1$$
  $$P(x = 1) = 0{,}1$$
  $$P(x = 2) = 0{,}2$$
  $$P(x = 3) = 0{,}3$$
  $$P(x = 4) = 0{,}2$$
  $$P(x = 5) = 0{,}1$$

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 1

Benutzungsprofil 2

Im zweiten Benutzungsprofil werden mehr dynamische textuelle Inhalte abgerufen, wie beispielsweise in einem Wiki. Weiterhin werden mehr Benutzer erwartet.

- Ankunftsrate 1 Benutzer pro Sekunde
- In 20% der Fälle werden statische HTML Seiten abgefragt, in 80% der Fälle dynamische.
- Es werden mehr dynamische Multimediainhalte abgefragt. Multimediainhalte sind nun zu 60% statisch und zu 40% dynamisch generiert.
- Es werden weniger Multimediainhalte pro Seite abgefragt, im Mittel nur 1,4.

  Die Verteilungen werden nicht direkt benötigt, nur ggf. für die Abschätzung in Entwurfsalternative 5: Thread Pool:
  Verteilung der Anzahl von Multimediaobjekten x pro Seite:

$$P(x = 0) = 0{,}3$$

$$P(x = 1) = 0{,}3$$

$$P(x = 2) = 0{,}2$$

$$P(x = 3) = 0{,}1$$

$$P(x = 4) = 0{,}1$$

Zeitstempel auf Extrablatt angeben: Benutzungsprofil 2

Ressourcenumgebung

Alle Komponenten sind auf einem Server eingesetzt, wie Abbildung 1 bereits zeigt. Die CPU des Systems ist ein AMD Athlon XP mit einer Taktrate von 1,145 GHz. Ein CPU Zyklus entspricht einer Instruktion. Für die Angaben wurde der Rechenaufwand in Work Units angegeben. Ein Work Unit entspricht ca. $1{,}145 * 10^6$ Instruktionen = $1{,}145 * 10^6$ CPU Zyklen. Es wurde gemessen, dass die Latenzzeit der Festplatte (Seek Time) 3,5 ms ist und sie weiterhin ca. 24 MB/s = $24 * 10^6$ Byte/s lesen bzw. schreiben kann.

Verwenden Sie für SPE-ED die Webserver Facility und legen Sie ein neues Software Spec Template an. Ändern Sie die auch Service Unit und Service Time der einzelnen Devices. Für die CPU bietet sich z.B. als Service Einheit „Megainstruktionen“ an.

Zeitstempel auf Extrablatt angeben: Ressourcenumgebung

Entwurfsalternativen

Modellieren Sie die Entwurfsalternativen erst, wenn das Ausgangsystem vollständig modelliert ist. Lassen Sie Ihr Ausgangssystem erst überprüfen, bevor Sie die Entwurfsalternativen modellieren.

Entwurfsalternative 1: Cache für dynamische Inhalte

Motivation

Die Erzeugung von Seiten mit dynamischem Inhalt erfordert Rechenzeit und muss im Normalbetrieb für jeden Request neu geschehen. Bei der Erzeugung von PHP Seiten muss beispielsweise zunächst das PHP Skript interpretiert werden und danach mit dem erzeugten ausführbaren PHP Code die tatsächliche HTML Seite generiert werden. Wenn jedoch dieselben Inhalte immer wieder angefragt werden, die außerdem rechenintensiv in der Erzeugung sind, kann es sinnvoll sein, einen Cache für dynamische Seiten einzusetzen. Ein Beispiel ist eAccelerator, einem sog. PHP Accelerator, der den ausführbaren PHP Code zwischenspeichert.

Anpassung des Systems

Die Komponente ContentGenerator wird in dieser Entwurfsalternative durch die Komponente AcceleratedContentGenerator ersetzt. Innerhalb dieser Komponente ist der Resource Demand zum Erzeugen eines dynamischen Inhalts nun anders als oben unter Ressourcenumgebung angegeben:

- 30 % des dynamischen Inhalts (Seiten und Multimediainhalte) liegen im Cache der AcceleratedContentGenerator Komponente bereits ausführbar vor. Für diese fallen nur 3 Work Units pro KB der resultierenden Datei für die Ausführung an.
- Für die anderen 70 % des dynamischen Inhalts wird weiterhin zuerst die Skriptdatei von der AcceleratedContentGenerator Komponente interpretiert und dann die Inhalte (HTML oder Multimediaobjekt) generiert. Der Aufwand zur Erzeugung liegt hier weiterhin bei 10 bzw. 15 Work Units pro KB der resultierenden Datei.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 1

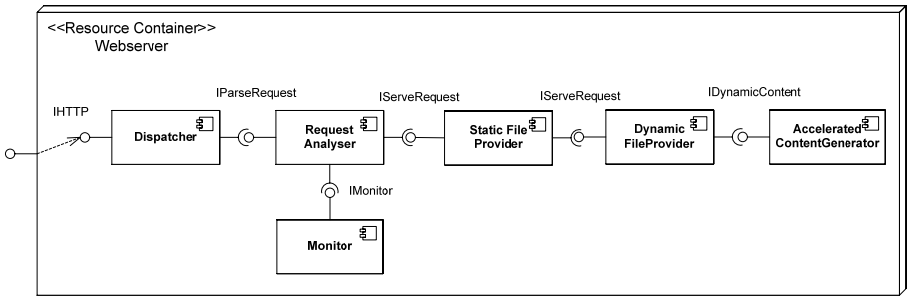


Abbildung 4: Entwurfsalternative 1: System und Deployment

Entwurfsalternative 2: Dynamischer Lookup

Motivation

Diese Entwurfsalternative hat im Gegensatz zu den bisherigen Alternativen nicht zum Ziel, die Performanz des Systems zu verbessern, sondern die Skalierbarkeit. Wenn alle Komponenten ihr Kommunikationspartner über einen dynamischen Lookup finden, kann die Allokation der Komponenten leicht verändert werden, beispielsweise können neue Server angeschafft und einige der Komponenten darauf eingesetzt werden, ohne Änderungen am Code vornehmen zu müssen. Allerdings benötigt das dynamische Auffinden des Kommunikationspartners zusätzlichen Rechenaufwand.

Es wurde beschlossen, dass der dynamische Lookup nur verwendet werden soll, wenn sein Einsatz die Antwortzeit des Systems für die gegebenen Benutzungsprofile um nicht mehr als 10% verschlechtert. Analysieren Sie, ob dieses Performanzziel eingehalten werden kann und geben Sie an, ob diese Entwurfsalternative nach diesen Vorgaben eingesetzt werden kann.

Angaben zum Rechenaufwand

- Der dynamische Lookup einer anderen Komponente kostet 17 Work Units.

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 2

Entwurfsalternative 3: Logging parallelisieren

Motivation

Eine weitere Alternative zur Reduzierung des Aufwands, das Logging über die Monitor-Komponente wenn möglich parallel zum eigentlichen Kontrollfluss auszuführen.

Anpassung des Systems

Dafür wird eine neue Komponente FastRequestAnalyser eingeführt, die die RequestAnalyser Komponente ersetzt und sich wie unten angeben verhält. Der erste Aufruf der Monitor-Komponente kann parallel zur Weiterverarbeitung des Requests durch den Static- bzw. DynamicFileProvider geschehen. Für den zweiten Aufruf werden die Ergebnisse des StaticFileProviders für die Zusammenfassung der Response benötigt. Der zweite Aufruf kann also nur parallel zur Generierung der Response durchgeführt werden. Um dies zu verdeutlichen, wurde im unten angegebenen Sequenzdiagramm die Generierung der HTTP Response diesmal explizit modelliert.

Um parallel arbeiten zu können, muss ein Thread für das Logging vom Haupt-Thread abgespalten werden, der Resource Demand dafür ist im Abschnitt Resource Demands auf Seite 6 angegeben. Dieser Thread kann für den zweiten parallelen Abschnitt wiederverwendet werden, der Aufwand für das Erzeugen eines neuen Threads fällt also nur einmal an.

In SPE-ED können Sie die Parallelität nicht direkt modellieren. Sie müssen hier abschätzen, welche Software Resource Requirements weggelassen werden können.

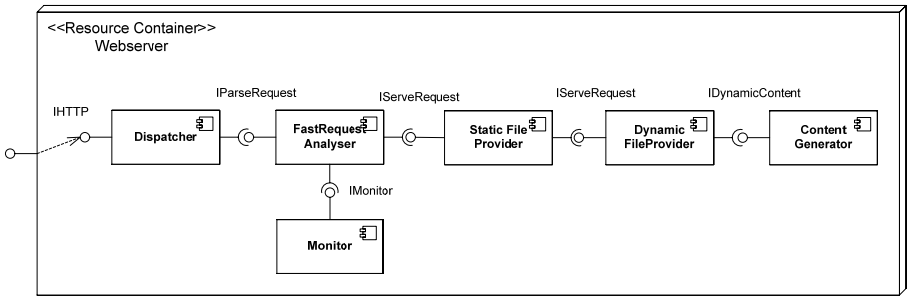


Abbildung 5: Entwurfsalternative 3: System und Deployment

Sequenzdiagramm auf der nächsten Seite.

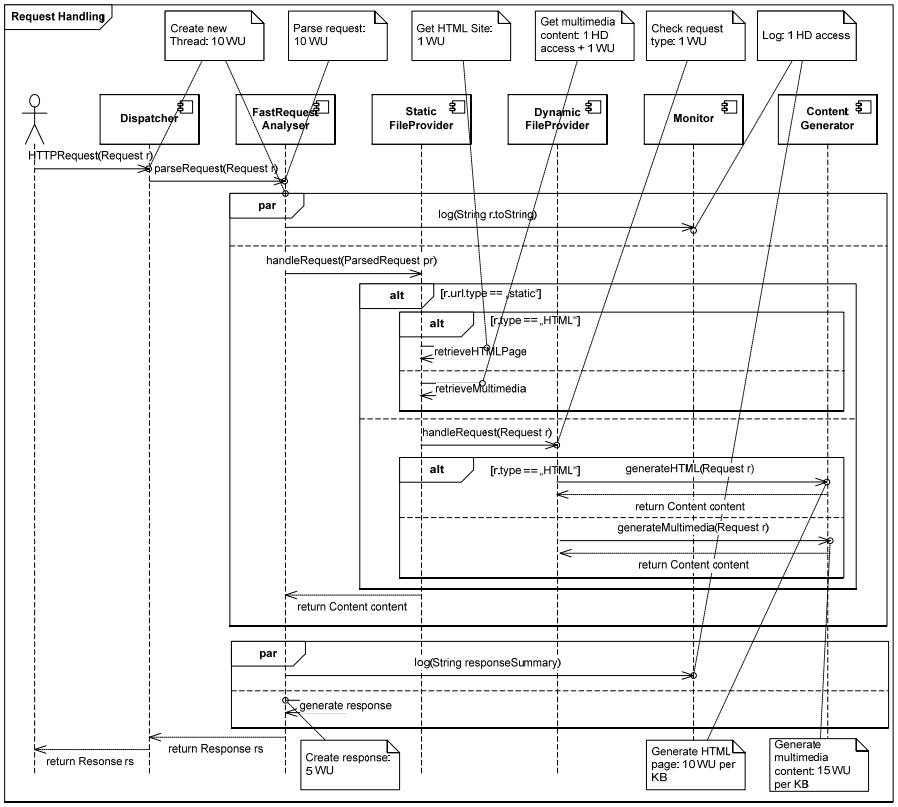


Abbildung 6: Entwurfsalternative 3: Veränderter Ablauf

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 3

Entwurfsalternative 4: Hardware Replikation mit einem zweiten Server

Motivation

Ein Teil der Komponenten wird in dieser Entwurfsalternative auf einen zweiten Server ausgelagert. In diesem Fall wurde beschlossen, die Komponenten DynamicFileProvider und ContentGenerator auf den zweiten Server auszulagern, so dass der erste Server die Anfragen annimmt, analysiert und statische Anfragen bedient, und der zweite dynamische Anfragen.

Angaben zur Ressourcenumgebung

Der zweite Server (AppServer) hat eine doppelt so schnelle CPU wie der erste. Die Festplatte hat die gleichen Leistungsmerkmale. Verwenden Sie das DB Server Facility Template und passen Sie die Daten in der Overhead Matrix an. Denken Sie daran, die Anzahl der Festplatten auf 1 zu setzen.

Das folgende Diagramm zeigt die neue Allokation des Systems:

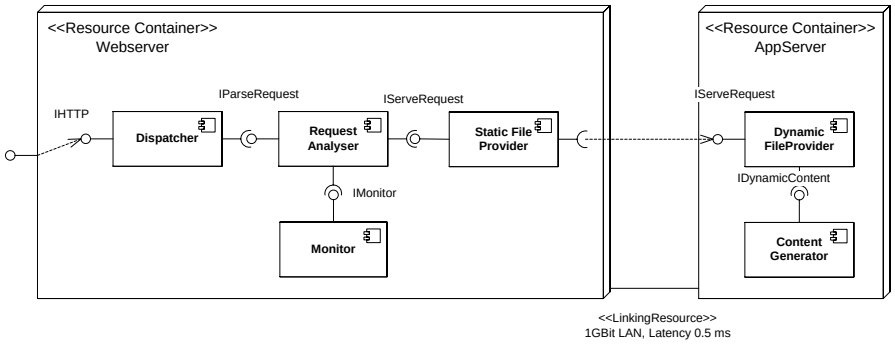


Abbildung 7: Entwurfsalternative 4: System und Deployment

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 4

Entwurfsalternative 5: Thread Pool

Motivation

Im Normalbetrieb erstellt der Webserver in der Dispatcher Komponente für jede Anfrage einen neuen Thread. Die Erzeugung eines Threads ist dabei mit Rechenaufwand verbunden.

Eine Entwurfsalternative ist die Einführung eines Thread Pools. Hier hält der Webserver bereits eine Anzahl Threads vor, und weist jedem Request dann nur noch einen Thread zu. Ein Vorteil dieser Entwurfsalternative ist, dass bei einer hohen Last nicht mehr zu viele Threads gleichzeitig aktiv sind und durch ständigen Kontextwechsel die Ressourcen nur suboptimal ausnutzen. Weiterhin wird die Zeit für die Erzeugung eines neuen Threads eingespart.

Anpassung des Systems

Modellieren Sie einen Threadpool der Größe 8. Gehen Sie dabei davon aus, dass mit dem Holen eines Threads aus dem Threadpool kein Rechenaufwand verbunden ist. Für SPE-ED müssen Sie abschätzen, wie lange die Anfrage eines Benutzers ggf. durch die Verwendung des Pools verzögert wird. Dies hängt von der Ankunftsrate der Benutzer sowie der Dauer, wie lange die Ressource blockiert ist, ab. Im Ein-Benutzer-Fall im Benutzungsprofil 1 tritt natürlich keine Verzögerung auf.

Geben Sie Ihre Schätzung sowie eine knappe Herleitung außerdem hier an: Durchschnittliche Verzögerung durch Thread Pool im Benutzungsprofil 2:

_____ weil

Diese Entwurfsalternative wird durch eine neue Komponente PoolingDispatcher realisiert, die die Komponente Dispatcher ersetzt.

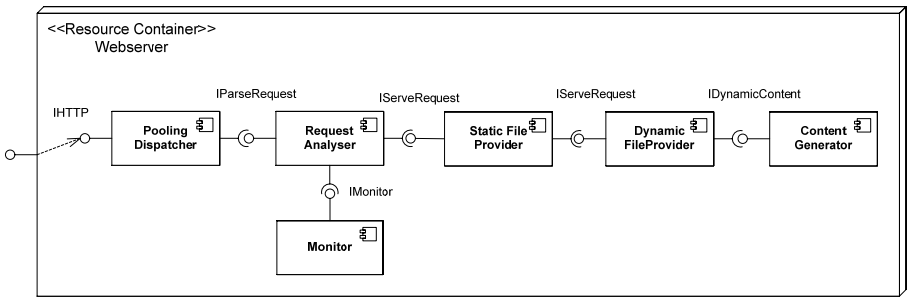


Abbildung 8: Entwurfsalternative 5: System und Deployment

Zeitstempel auf Extrablatt angeben: Entwurfsalternative 5

Fragestellungen

Untersuchen Sie das Ausgangssystem und die verschiedenen Entwurfsalternativen und sichern Sie die Ergebnisse wie oben unter Durchführung (Seite 2) angegeben.

1. Welche Entwurfsalternativen sollten eingesetzt werden*, welche nicht? Differenzieren Sie Ihre Antwort nach den beiden Benutzungsprofilen. Geben Sie Ihre Antwort und eine Begründung in der Tabelle 1 unten an.
   (*: Die Alternative 2 braucht die Antwortzeit nicht verbessern, darf sie aber nicht mehr als 10% verschlechtern, um eingesetzt zu werden.)
2. Angenommen, es wäre nicht möglich, alle sinnvollen Entwurfsalternativen umzusetzen, sondern nur einige, welche wäre das? Stellen Sie eine Reihenfolge der Entwurfsentscheidungen auf, um diese Frage zu beantworten, und sortieren Sie die Entwurfsentscheidungen in Tabelle 2, beginnend mit der nützlichsten. Beachten Sie dabei nicht eventuelle Abhängigkeiten der Entwurfsentscheidungen untereinander.

Verschiedene Zeitstempel auf Extrablatt angeben: Analyse

Entwurfsalternative	Benutz-ungs-profil	Durchschn. Antwort-zeit	Einsetzen? Mit Begründung.
1: Cache für dyn. Inhalte	1		
	2		
2: Dynamischer Lookup	1		
	2		
3: Logging parallelisieren	1		
	2		
4: Hardware Replikation	1		
	2		
5: Thread Pool	1		
	2		

Tabelle 1: Bewertung der Entwurfsalternativen

	Reihenfolge	
Entwurfsalternative	**Benutzungsprofil 1**	**Benutzungsprofil 2**
1: Cache für dyn. Inhalte		
3: Logging parallelisieren		
4: Hardware Replikation		
5: Thread Pool		

Tabelle 2: Reihenfolge der Entwurfsalternativen

Kommentare zu den Entwurfsalternativen:

Applying Bayes' Rule

If the control flow for both kinds of requests are combined, the arrival rate needs adjustment and the probabilities of the different branches have to be calculated using Bayes' Rule (for details, see [Sac97, p.78]). In the task description, the following probabilities are given for usage profile 1:

$$p(HTML) = 1/3.5, \;\; p(Multimedia) = 2.5/3.5$$

$$p(static|HTML) = 0.4, \;\; p(dynamic|HTML) = 0.6$$

$$p(static|Multimedia) = 0.7, \;\; p(dynamic|Multimedia) = 0.3$$

For usage profile 2, these probabilities are given:

$$p(HTML) = 1/2.4, \;\; p(Multimedia) = 1.4/2.4$$

$$p(static|HTML) = 0.2, \;\; p(dynamic|HTML) = 0.8$$

$$p(static|Multimedia) = 0.6, \;\; p(dynamic|Multimedia) = 0.4$$

For the modelling of the control flow, however, it first needs to be distinguished whether a static or dynamic object is requested, because the responsible component is found out first. Only after this the respective component handles the distinction of HTML and Multimedia contents. Thus, for the annotations of the first branch to find the responsible component, the probabilities $p(static)$ and $p(dynamic)$ are needed. They can easily be calculated from the probability tree.

$$\begin{aligned} p(static) &= p(HTML)p(static|HTML) + p(Multimedia)p(static|Multimedia) \\ &= 0.614 \\ p(dynamic) &= p(HTML)p(dynamic|HTML) + p(Multimedia)p(dynamic|Multimedia) \\ &= 0.386 \end{aligned}$$

For usage profile 2, the following values are calculated:

$$\begin{aligned} p(static) &= 0.433 \\ p(dynamic) &= 0.567 \end{aligned}$$

Now, for the responsible component the probabilities whether an HTML or a Multimedia object is requested are needed, namely $p(HTML|static)$, $p(Multimedia|static)$, $p(HTML|dynamic)$ and $p(Multimedia|dynamic)$. These values can be calculated by applying Bayes' Rule, which helps to calculated a posteriori probabilities:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Using Bayes' Rule, the needed probabilities can be calculated as follows:

$$\begin{aligned} p(HTML|static) &= \frac{p(static|HTML) \cdot p(HTML)}{p(static)} = 0.186 \\ p(Multimedia|static) &= \frac{p(static|Multimedia) \cdot p(Multimedia)}{p(static)} = 0.814 \\ p(HTML|dynamic) &= \frac{p(dynamic|HTML) \cdot p(HTML)}{p(dynamic)} = 0.444 \\ p(Multimedia|dynamic) &= \frac{p(dynamic|Multimedia) \cdot p(Multimedia)}{p(dynamic)} = 0.556 \end{aligned}$$

For usage profile 2:

$$\begin{aligned} p(HTML|static) &= 0.192 \\ p(Multimedia|static) &= 0.808 \\ p(HTML|dynamic) &= 0.588 \\ p(Multimedia|dynamic) &= 0.412 \end{aligned}$$

B.2 Rank Estimation

B.2.1 Media Store

Abschätzung des Nutzens der Entwurfsalternativen

Name: _____

Bitte geben Sie eine Abschätzung ab, welche Entwurfsalternative die Antwortzeit am stärksten verbessert, welche am zweitstärksten, usw. Stellen Sie dafür eine Reihenfolge auf, beginnend mit der besten Entwurfsalternative. Sie können einen Rang zweimal vergeben, wenn Sie zwei Alternativen für gleich gut halten. Geben Sie außerdem an, ob die Entwurfsalternativen die Performanz überhaupt positiv beeinflussen werden und wie stark.

Verwenden Sie für die Einschätzung folgende Abkürzungen:

- - Verschlechtert die Performanz deutlich
- - Verschlechtert die Performanz etwas
- o Neutral
- \+ Verbessert die Performanz etwas
- ++ Verbessert die Performanz deutlich

	Reihenfolge		**Verbessert Antwortverhalten?**	
Entwurfsalternative	Ein-benutzerfall	Mehr-benutzerfall	Ein-benutzerfall	Mehr-benutzerfall
1: Cache				
2: Pool für Datenbank-verbindungen				
3: Zweiter Server				
4: Bitrate senken				
5: Dynamischer Lookup				

Kommentare:

B.2.2 Web Server

Abschätzung des Nutzens der Entwurfsalternativen

Name: _____

Bitte geben Sie eine Abschätzung ab, welche Entwurfsalternative die Antwortzeit am stärksten verbessert, welche am zweitstärksten, usw. Stellen Sie dafür eine Reihenfolge auf, beginnend mit der besten Entwurfsalternative. Sie können einen Rang zweimal vergeben, wenn Sie zwei Alternativen für gleich gut halten. Geben Sie außerdem an, ob die Entwurfsalternativen die Performanz überhaupt positiv beeinflussen werden und wie stark.

Verwenden Sie für die Einschätzung folgende Abkürzungen:

- - Verschlechtert die Performanz deutlich
- - Verschlechtert die Performanz etwas
- o Neutral
- \+ Verbessert die Performanz etwas
- ++ Verbessert die Performanz deutlich

	Reihenfolge		**Verbessert Antwortverhalten?**	
Entwurfsalternative	Ein-benutzerfall	Mehr-benutzerfall	Ein-benutzerfall	Mehr-benutzerfall
1: Cache für dynamische Inhalte				
2: Dynamischer Lookup				
3: Logging parallelisieren				
4: Hardware-Replikation mit zweitem Server				
5: Thread Pool				

Kommentare:

B.3 Time Stamps

B.3.1 Media Store

Palladio

Zeiten Palladio

Name: _____

Die Teilbereiche müssen nicht in dieser Reihenfolge bearbeitet werden, deshalb sind die Zeiten hier nicht unbedingt chronologisch geordnet. Geben Sie bitte für jeden Aufgabe Start- und Endzeitpunkt an. Wenn Sie zwei Aufgaben parallel bearbeiten, geben Sie dies bitte an.

Austeilung der Aufgabenstellung: _____

Kontrollfluss

Durchgelesen: _____

Modelliert: _____

Resource Demand

Durchgelesen: _____

Modelliert: _____

Ressourcenumgebung

Durchgelesen: _____

Modelliert: _____

Benutzungsprofil 1

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Benutzungsprofil 2

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 1

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 2

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 3

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 4

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 5

Durchgelesen: _____

Modelliert: _____

Fehlersuche: _____

Analyse: _____

Bestimmung der Reihenfolge der Entwurfsalternativen: _____

Zusatzfrage: _____

Abgabe: _____

SPE

Zeiten SPE-ED

Name: _____

Die Teilbereiche müssen nicht in dieser Reihenfolge bearbeitet werden, deshalb sind die Zeiten hier nicht unbedingt chronologisch geordnet. Geben Sie bitte für jeden Aufgabe Start- und Endzeitpunkt an. Wenn Sie zwei Aufgaben parallel bearbeiten, geben Sie dies bitte an.

Austeilung der Aufgabenstellung: _____

Kontrollfluss

Durchgelesen: _____

Modelliert: _____

Ressourcenumgebung

Durchgelesen: _____

Modelliert: _____

Resource Demand und Benutzungsprofile

Resource Demand durchgelesen: _____

Benutzungsprofil 1 durchgelesen: _____

Mit Benutzungsprofil 1 modelliert: _____

Fehlersuche: _____

Analyse: _____

Benutzungsprofil 2 durchgelesen: _____

Mit Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 1

Durchgelesen: _____

Für Benutzungsprofil 1 modelliert: _____

Für Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 2

Durchgelesen: _____

Für Benutzungsprofil 1 modelliert: _____

Für Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 3

Durchgelesen: _____

Für Benutzungsprofil 1 modelliert: _____

Für Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 4

Durchgelesen: _____

Für Benutzungsprofil 1 modelliert: _____

Für Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Entwurfsalternative 5

Durchgelesen: _____

Für Benutzungsprofil 1 modelliert: _____

Für Benutzungsprofil 2 modelliert: _____

Fehlersuche: _____

Analyse: _____

Bestimmung der Reihenfolge der Entwurfsalternativen: _____

Zusatzfrage: _____

Abgabe: _____

B.3.2 Web Server

Palladio

Zeiten Palladio

Name: _____

Die Teilbereiche müssen nicht in dieser Reihenfolge bearbeitet werden, deshalb sind die Zeiten hier nicht unbedingt chronologisch geordnet. Geben Sie bitte für jeden Aufgabe Start- und Endzeitpunkt oder die Dauer an. Wenn Sie zwei Aufgaben parallel bearbeiten, schätzen Sie bitte die Anteile ab, und geben Sie die Dauer einzeln an. Insgesamt sollten die Zeiten so angegeben werden, dass wir die Dauer der einzelnen Aufgabenteile bestimmen können.

Austeilung der Aufgabenstellung: _____

	Startzeitpunkt	**Endzeitpunkt**	**Dauer**
Durchlesen			
Abschätzung und Reihenfolge im Voraus			

Kontrollfluss			
Modelliert:			

Resource Demand			
Modelliert:			

Ressourcenumgebung			
Modelliert			

Benutzungsprofil 1			
Modelliert			
Fehlersuche			
Analyse			

Benutzungsprofil 2			
Modelliert			
Fehlersuche			
Analyse			

	Startzeitpunkt	**Endzeitpunkt**	**Dauer**
Entwurfsalternative 1			
Modelliert			
Fehlersuche			
Analyse für Benutzungsprofil 1			
Analyse für Benutzungsprofil 2			

Entwurfsalternative 2			
Modelliert			
Fehlersuche			
Analyse für Benutzungsprofil 1			
Analyse für Benutzungsprofil 2			

Entwurfsalternative 3			
Modelliert			
Fehlersuche			
Analyse für Benutzungsprofil 1			
Analyse für Benutzungsprofil 2			

Entwurfsalternative 4			
Modelliert			
Fehlersuche			
Analyse für Benutzungsprofil 1			
Analyse für Benutzungsprofil 2			

Entwurfsalternative 5			
Modelliert			
Fehlersuche			
Analyse für Benutzungsprofil 1			
Analyse für Benutzungsprofil 2			

Bestimmung der Reihenfolge der Entwurfsalternativen:			

Abgabe: _____

SPE

Zeiten SPE-ED

Name: _____

Die Teilbereiche müssen nicht in dieser Reihenfolge bearbeitet werden, deshalb sind die Zeiten hier nicht unbedingt chronologisch geordnet. Geben Sie bitte für jeden Aufgabe Start- und Endzeitpunkt oder die Dauer an. Wenn Sie zwei Aufgaben parallel bearbeiten, schätzen Sie bitte die Anteile ab, und geben Sie die Dauer einzeln an. Insgesamt sollten die Zeiten so angegeben werden, dass wir die Dauer der einzelnen Aufgabenteile bestimmen können.

Austeilung der Aufgabenstellung: _____

	Startzeitpunkt	**Endzeitpunkt**	**Dauer**
Durchlesen			
Abschätzung Voraus			

Kontrollfluss			
Modelliert:			

Ressourcenumgebung			
Modelliert			

Mit Benutzungsprofil 1 modelliert			
Fehlersuche			
Analyse:			

Mit Benutzungsprofil 2 modelliert			
Fehlersuche			
Analyse			

Entwurfsalternative 1			
Für Benutzungsprofil 1 modelliert:			
Für Benutzungsprofil 2 modelliert:			
Fehlersuche:			
Analyse:			

	Startzeitpunkt	**Endzeitpunkt**	**Dauer**
Entwurfsalternative 2			
Für Benutzungsprofil 1 modelliert:			
Für Benutzungsprofil 2 modelliert:			
Fehlersuche:			
Analyse:			

Entwurfsalternative 3			
Für Benutzungsprofil 1 modelliert:			
Für Benutzungsprofil 2 modelliert:			
Fehlersuche:			
Analyse:			

Entwurfsalternative 4			
Für Benutzungsprofil 1 modelliert:			
Für Benutzungsprofil 2 modelliert:			
Fehlersuche:			
Analyse:			

Entwurfsalternative 5			
Für Benutzungsprofil 1 modelliert:			
Für Benutzungsprofil 2 modelliert:			
Fehlersuche:			
Analyse:			

Bestimmung der Reihenfolge der Entwurfsalternativen:			

Abgabe: _____

B.4 Qualitative Questionnaires

B.4.1 Questionnaire Media Store

Fragebogen zur Mediastore-Aufgabe

Name: _____ Verwendetes Werkzeug: _____

Aufgabenstellung und Methode

Bitte geben Sie an, welche Teile der Aufgabenstellung verständlich waren und welche nicht. Was waren die Probleme bei der Besarbeitung? Fügen Sie ggf. Aspekte hinzu.

	Verständ-lich?	Probleme
Aufgabe im Ganzen		
Statische Struktur des Systems		
Kontrollfluss		
Resource Demands		
Benutzungsprofil		

Welcher Aspekt der Methode war am schwierigsten umzusetzen? Was war das Problem?

War die Bearbeitungszeit angemessen? Falls nicht, war sie zu lang oder zu kurz?

Wofür haben Sie die meiste Zeit benötigt? Wie lange in etwa?

Welche anderen Aufgaben(-teile) waren zeitaufwändig? Wie lange haben Sie in etwa dafür benötigt?

Vorbereitung

Wurden Sie im Praktikum auf alle Aspekte der Aufgabe vorbereitet? Falls nicht, welche Aspekte fehlten?

Welche Aspekte wurden zwar im Praktikum vorgestellt, allerdings nicht in der Tiefe, wie sie für die Aufgabe benötigt wurde?

Werkzeug

Traten während der Bearbeitung der Aufgabe Probleme mit dem Werkzeug auf? Wenn ja, welche?

B.4.2 Questionnaire Web Server

Fragebogen zur Webserver-Aufgabe

Name: _____ Verwendetes Werkzeug: _____

Aufgabenstellung und Methode

Bitte geben Sie an, welche Teile der Aufgabenstellung verständlich waren und welche nicht. Was waren die Probleme bei der Bearbeitung? Fügen Sie ggf. Aspekte hinzu.

	Verständ-lich?	Probleme
Aufgabe im Ganzen		
Statische Struktur des Systems		
Kontrollfluss		
Resource Demands		
Benutzungsprofil		

Welcher Aspekt der Methode war am schwierigsten umzusetzen? Was war das Problem?

War die Bearbeitungszeit angemessen? Falls nicht, war sie zu lang oder zu kurz?

Wofür haben Sie die meiste Zeit benötigt? Wie lange in etwa?

Welche anderen Aufgaben(-teile) waren zeitaufwändig? Wie lange haben Sie in etwa dafür benötigt?

Vorbereitung

Wurden Sie im Praktikum auf alle Aspekte der Aufgabe vorbereitet? Falls nicht, welche Aspekte fehlten?

Welche Aspekte wurden zwar im Praktikum vorgestellt, allerdings nicht in der Tiefe, wie sie für die Aufgabe benötigt wurde?

Werkzeug

Traten während der Bearbeitung der Aufgabe Probleme mit dem Werkzeug auf? Wenn ja, welche?

B.4.3 Comparing Questionnaire

Nachträglicher Fragebogen

Name: _____

Dieser Fragebogen dient nicht dazu, Sie zu bewerten, sondern die Verfahren. Ihre Antworten haben also keinerlei Auswirkung auf die Benotung.

Aufgabenstellungen

1. Waren die Aufgabenstellung 1 (Mediastore) Ihrem Kenntnisstand angemessen? Wenn nicht, was war problematisch?

2. Waren die Aufgabenstellung 2 (Webserver) Ihrem Kenntnisstand angemessen? Wenn nicht, was war problematisch?

3. War der Schwierigkeitsgrad der Aufgaben unterschiedlich? Falls ja, welche war schwerer und warum?

4. Konnten Sie Erfahrungen aus der ersten Aufgabenstellung in der zweiten anwenden? Wenn ja, welche?

Palladio

Konzepte

5. Ist das Vorgehensmodell verständlich? Falls nicht, was war problematisch?

6. Konnte das Metamodell nachvollzogen werden? Falls nicht, was war problematisch?

7. Bewerten Sie die Verständlichkeit der folgenden Konzepte. Bitte bewerten Sie die Aspekte auf einer Skala ++, +, o, -, --. Bitte bewerten Sie das Konzept an sich und nicht die Umsetzung in den Werkzeugen.

Was sind die Gründe für Ihre Bewertung?

Konzept	**Bewertung**	**Gründe**
Repository Model		
SEFF Spezifikation		
System		
Allocation		
Resource Environment		
Usage Model		
Parametri-sierung		
Visualisierung der Ergebnisse		
Verteilungs-funktionen		

8. Hilft die Rollenaufteilung beim Verständnis der verschiedenen Konzepte?

9. Bewerten Sie die Parametrisierung der SEFFs und die getrennte Spezifikation der Parameter im Usage Model und als Komponentenparameter. Welche Vorteile und Nachteile sehen Sie?

10. Sehen Sie weitere oder andere Vor- bzw. Nachteile der Parametrisierung bei größeren, komplexeren Modellen?

11. Erleichtert die Parametrisierung die Spezifikation eines SEFF an sich oder bedeutet sie einen Mehraufwand?

12. Erleichtert oder erschwert die Parametrisierung die Spezifikation von komplexen Verzweigungswahrscheinlichkeiten, wie beispielsweise in der Bitraten-Alternative des Mediastores oder bei der Bestimmung der Art des gelieferten Inhalts beim Webserver?

13. Bewerten Sie automatisierte Transformationen, die übliche performanzrelevante Entscheidungen einbauen (denken Sie an die Broker-Alternative mit dem dynamischen Lookup von Komponenten). Denken Sie auch an Fälle, die komplexer als unser Broker Beispiel wären. Welche Vorteile und Nachteile sehen Sie?

Werkzeug

14. Ist das PCM Werkzeug geeignet für die Performanzvorhersage? Was sind positive, was negative Aspekte?

15. Würden Sie anstelle der konkreten graphischen Syntax für einige Teile eine textuelle Syntax bevorzugen, z.B. für den SEFF eine Art Pseudo-Code? Wenn ja, welche Teile des Modells würden Sie auf textuelle Eingabe umstellen?

16. Falls die Parametrisierung problematisch war: Ist es ein prinzipielles Problem oder liegt es an der Darstellung im SEFF bzw. Usage Model?

Verbesserung

17. Wenn Sie das Palladio Verfahren und sein Werkzeug verbessern könnten, was würden Sie ändern?

SPE

Konzepte

18. Ist das Vorgehensmodell verständlich?

19. Bewerten Sie die Verständlichkeit der folgenden Konzepte. Bitte bewerten Sie die Aspekte auf einer Skala ++, +, o, -, --. Bitte bewerten Sie das Konzept an sich und nicht die Umsetzung in den Werkzeugen.

    Was sind die Gründe für Ihre Bewertung?

Konzept	**Bewertung**	**Gründe**
Aufteilung in Scenarios		
Software Model		
Overhead Matrix		

Konzept	**Bewertung**	**Gründe**
System Model		
Verteilte Systeme		
Verschiedene Solutions		
Visualisierung der Ergebnisse		

Werkzeug

20. Ist das SPE-ED Werkzeug geeignet für die Performanzvorhersage? Was sind positive, was negative Aspekte?

Verbesserung

21. Wenn Sie das SPE Verfahren und sein Werkzeug SPE-ED verbessern könnten, was würden Sie ändern?

Interpretierung der Ergebnisse

22. War die Interpretation der Verteilungsfunktionen bei Palladio schwerer als die Mittelwertbetrachtung bei SPE-ED?

23. Liefert die Auswertung von Verteilungsfunktionen bessere Entscheidungsgrundlagen für Entwurfsentscheidungen?

Aufwand

24. Welches Verfahren hat den höheren Lernaufwand? Warum?

25. Welches Verfahren ist aufwändiger in der Anwendung? Warum?

26. Bei Palladio werden die Aufgaben eigentlich auf Rollen aufgeteilt, die Sie im Experiment alle gleichzeitig eingenommen haben. Wie bewerten Sie die Aufwandsfrage, wenn die Rollen von verschiedenen Personen ausgeführt worden wären?

27. Welches Werkzeug ist schneller zu bedienen?

Vergleich der Verfahren

28. Welches Verfahren ist besser zur Vorhersage der Performanz geeignet?

29. Haben Sie bei einem Verfahren ein größeres Vertrauen in die Güte der Vorhersagen? Wenn ja, bei welchem Verfahren und warum?

30. Welches der Verfahren ist besser verständlich?

31. Mit welchem Verfahren haben Sie lieber gearbeitet? Warum?

32. Welches Werkzeug bevorzugen Sie? Was sind die Gründe dafür?

Weitere Kommentare

Falls Sie noch weitere Kommentare haben, geben Sie sie hier an:

B.5 Acceptance Tests

B.5.1 Check Lists

Media Store

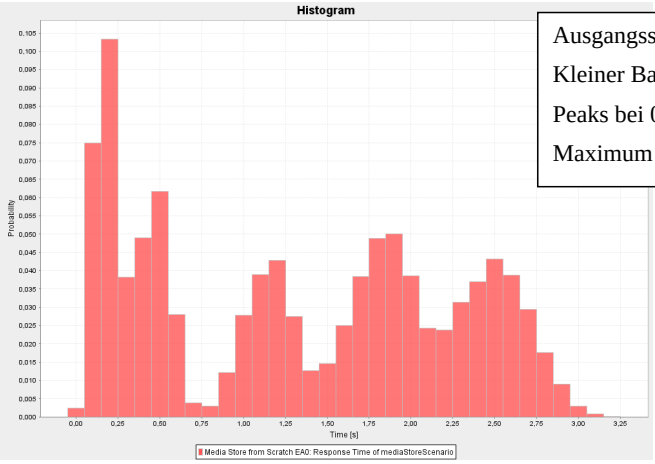
Checkliste

Palladio:

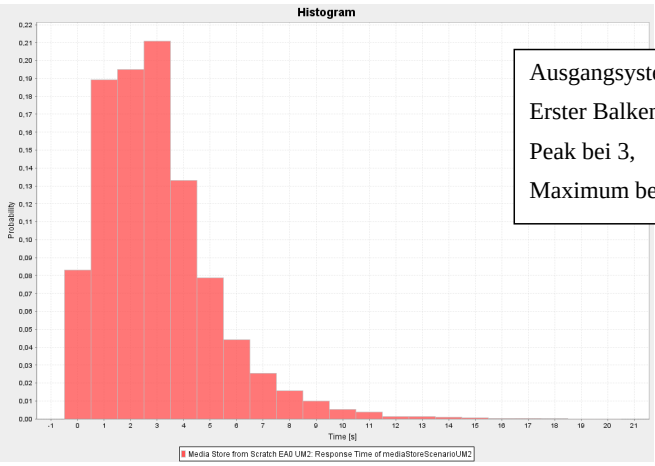
- Analysiert mit Simulation
- Ergebnisse im Rahmen
- Beide Anwendungsfälle modelliert
- Kontrollfluss durch 4 bzw. 3 Komponenten
- Processing Rate im Resource Environment gesetzt.

Entwurfsalternativen:

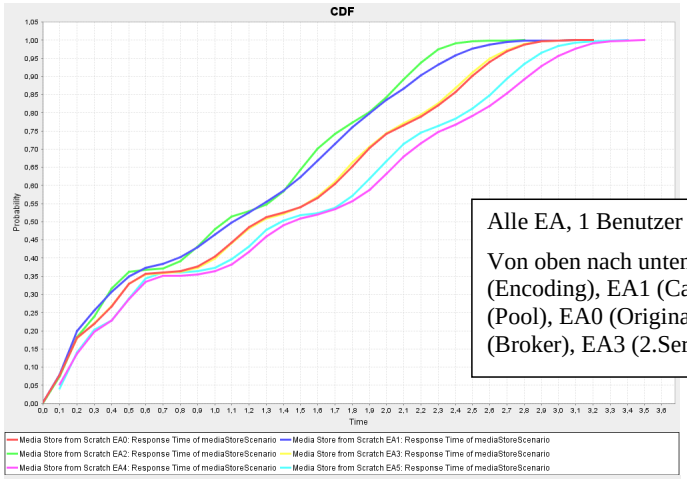
1. Branch bei Cache
2. Pool mit Acquire Release
3. Neue Allocation mit zweitem Server
4. Fallunterscheidung im Encoding SEFF
5. Haken für dynamischen Lookup



Ausgangssystem, 1 Benutzer
Kleiner Balken bei 0,
Peaks bei 0.2, 1.2,1.9,2.5,
Maximum bei 3.1

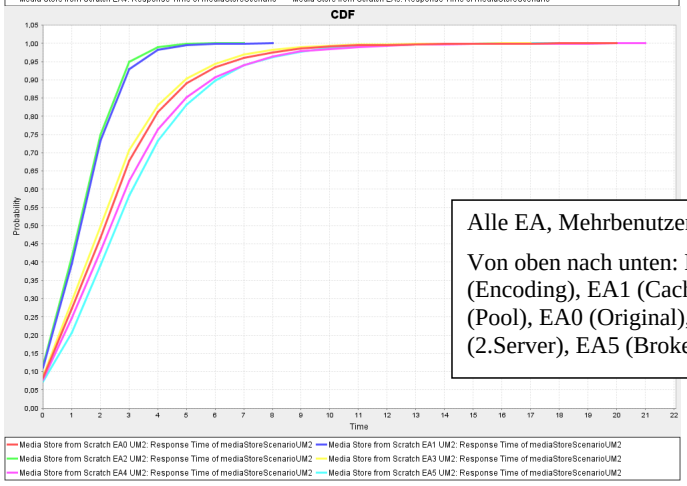


Ausgangsystem, Mehrbenutzer
Erster Balken bei 0,
Peak bei 3,
Maximum bei 15



Alle EA, 1 Benutzer

Von oben nach unten: EA4 (Encoding), EA1 (Caching), EA2 (Pool), EA0 (Original), EA5 (Broker), EA3 (2.Server)



Alle EA, Mehrbenutzer

Von oben nach unten: EA4 (Encoding), EA1 (Caching), EA2 (Pool), EA0 (Original), EA3 (2.Server), EA5 (Broker)

SPE:

- Analysiert mit Contention Solution (ggf. Simulation)?
- Ergebnisse im Rahmen
- Beide Anwendungsfälle modelliert
- Download mit Loop
- Overhead Matrix ausgefüllt

EA \ BP	1	2
Original	1.3317	8.9139
1: Cache	1.1627	3.952
2: Pool	1.3214	9.3835
3: 2. Server	1.352	8.9332
4: Bitrate	1.069	3.9575
5: Broker	1.4793	9.7487

Entwurfsalternativen:

1. Branch bei Cache
2. Abschätzung für Acquire Release angegeben auf dem Zettel
3. Zweites Szenario für zweiten Server
4. Fallunterscheidung für Encoding Alternativen
5. Zusätzliche Work Units für dynamischen Lookup

Web Server

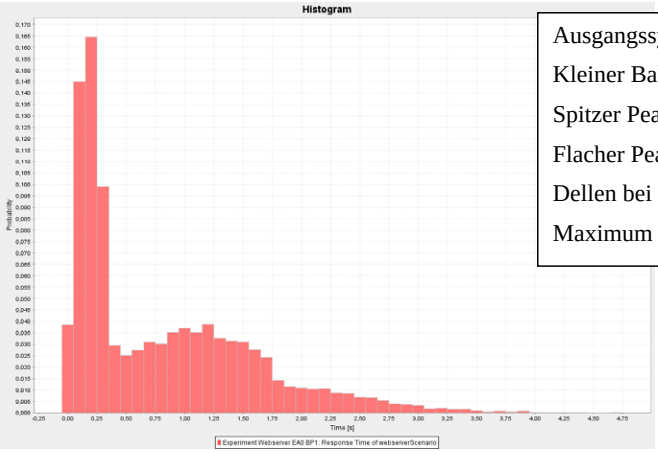
Checkliste Webserver

Palladio:

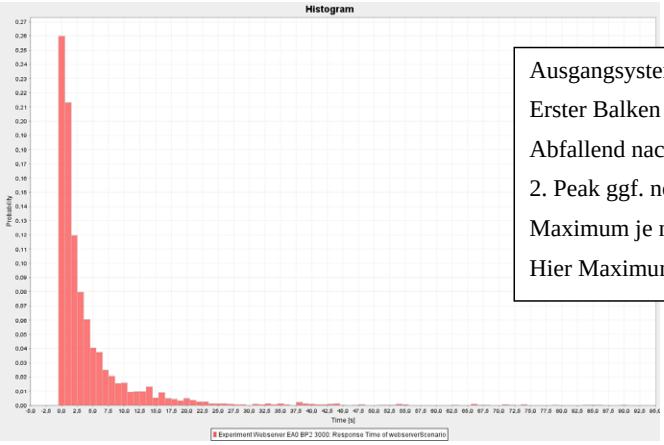
- Analysiert mit Simulation
- Ergebnisse im Rahmen
- Anwendungsfall mit beiden Aufrufen des Webservers modelliert
- Kontrollfluss durch 6 Komponenten
- Processing Rate im Resource Environment gesetzt.
- CDF und Histogramm mit Auflösung <1 Sek. betrachten.

Entwurfsalternativen:

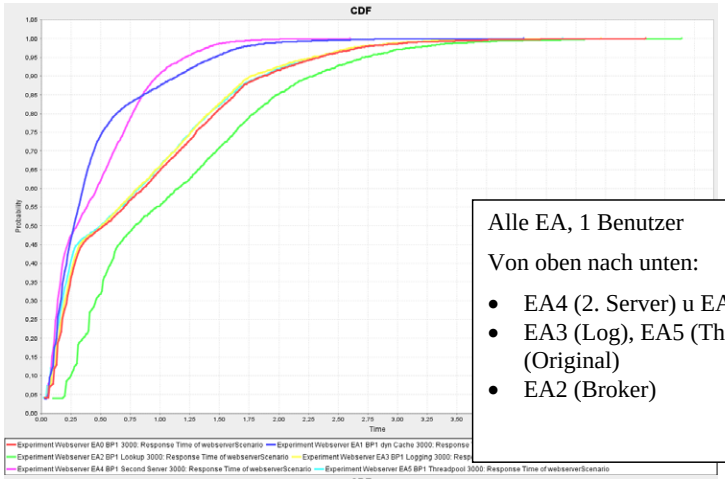
1. Branch bei Cache
2. Haken für dynamischen Lookup
3. 2 Fork Actions für Parallelität
4. Neue Allokation mit zweitem Resource Environment
5. Acquire-Release in Dispatcher Komponente



Ausgangssystem, 1 Benutzer
Kleiner Balken bei 0,
Spitzer Peak bei 0.2
Flacher Peak bei 1,2
Dellen bei 0,5 und 1,8
Maximum bei knapp unter 4



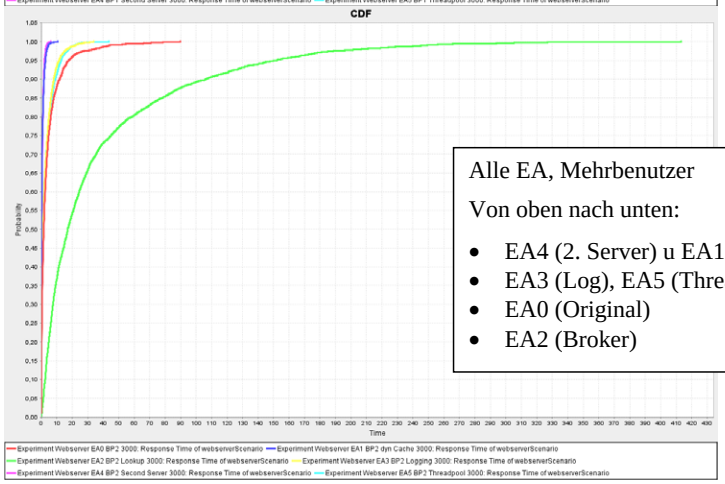
Ausgangsystem, Mehrbenutzer
Erster Balken bei 0,
Abfallend nach links
2. Peak ggf. noch zu erkennen
Maximum je nach SimTime
Hier Maximum 75 bei ST 3000



Alle EA, 1 Benutzer

Von oben nach unten:

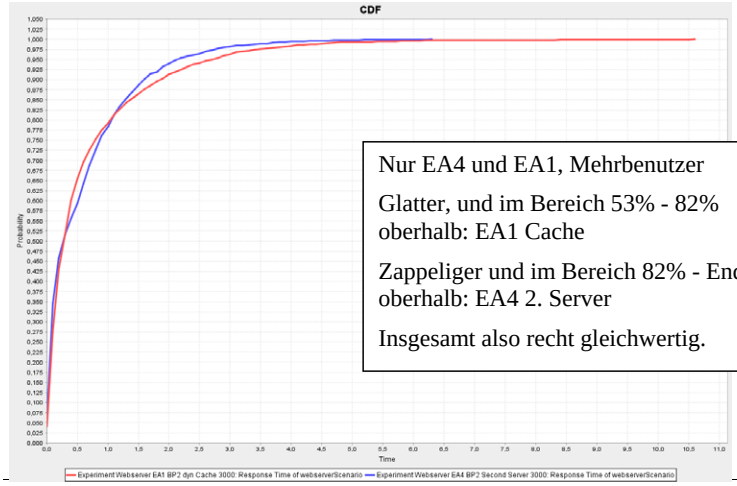
- EA4 (2. Server) u EA1 (Caching),
- EA3 (Log), EA5 (Threadpool), EA0 (Original)
- EA2 (Broker)



Alle EA, Mehrbenutzer

Von oben nach unten:

- EA4 (2. Server) u EA1 (Caching),
- EA3 (Log), EA5 (Threadpool)
- EA0 (Original)
- EA2 (Broker)



Nur EA4 und EA1, Mehrbenutzer

Glatter, und im Bereich 53% - 82% oberhalb: EA1 Cache

Zappeliger und im Bereich 82% - Ende oberhalb: EA4 2. Server

Insgesamt also recht gleichwertig.

SPE:

- Analysiert mit Contention Solution (ggf. Simulation)?
- Ergebnisse im Rahmen
- Beide Requests modelliert
- Overhead Matrix ausgefüllt

Entwurfsalternativen:

1. Branch bei Cache
2. Zusätzliche Work Units für dynamischen Lookup
3. HD Access bei Logger weggelassen
4. Verteilung in 2 Szenarien, mit Delay eingesetzt
5. Abschätzung für Acquire Release angegeben auf dem Zettel

EA \ BP	1	2
0: Original	0.7599	1.3941
1: Cache	0.5754	0.8652
2: Broker	1.0438	3.129
3: Par. Log	0.7703	1.3768
4: 2. Server	0.4425	*0.4033**
5: Pool	0.7249	1.2663

*: Komischer Wert, sollte eigentlich höher sein. Daher höhere Werte ok!

B.5.2 Acceptance Test Protocols

Akzeptanztest Webserver

Name des Teilnehmers: _____

Angeben, *wann* Ergebnisse geprüft wurden, und mit welchem *Ergebnis*. Falls Probleme auftraten, diese hier vermerken.

Stufen: abgenommen von:

1. Ausgangssystem mit Analyse

2. Entwurfsalternative 1

3. Entwurfsalternative 2

4. Entwurfsalternative 3

5. Entwurfsalternative 4

6. Entwurfsalternative 5

B.6 Question Protocol

B.6.1 Question Protocol Sheets

Media Store

Fragenprotokoll

Wer stellte *wann* welche *Frage*? Welches *Werkzeug* hat er benutzt? *Wer beantwortete* die Frage?

Web Server

Fragenprotokoll vom 07.07.2007

Fragen beantwortet von: _____

Wann	Wer	Werkzeug	Frage

Wann	Wer	Werkzeug	Frage

C Resulting Data

The models created by the participants can be found on the accompanying DVD of this thesis. If you are reading a printed version without DVD or an electronic version of this work, and are interested in the created models, please contact the author.

C.1 Predictions of the Participants

Predicted mean response time for the $UP1$ of Media Store with Palladio							
$p \in P_{Pal}^{MS}$	v_0^{MS}	v_1^{MS}	v_2^{MS}	v_3^{MS}	v_4^{MS}	v_5^{MS}	Mean deviation
1	1.32	1.16	1.31	1.33	1.15	1.45	0.04
2	1.31	1.13				1.44	0.03
3	1.30	1.09	1.31	1.47	1.10	1.41	0.05
4	1.28	1.09				1.43	0.06
5	1.34	1.16	1.32	1.49	1.20	1.46	0.02
6	1.36	1.18	1.32	2.48		1.48	0.20
7	1.31	1.14	1.32			1.47	0.01
Reference	1.33	1.16	1.32	1.52	1.15	1.47	0
Deviation	0.03	0.03	0.01	0.30	0.03	0.02	0.07

Predicted mean response time for the $UP2$ of Media Store with Palladio							
$p \in P_{Pal}^{MS}$	v_0^{MS}	v_1^{MS}	v_2^{MS}	v_3^{MS}	v_4^{MS}	v_5^{MS}	Mean deviation
1	3.14	1.87	2.89	3.01	1.76	3.70	0.18
2	2.92	1.86				3.41	0.23
3	2.73	1.78	2.90	3.18	1.71	3.54	0.19
4	2.78	1.76				3.28	0.32
5	3.10	1.86	3.01	3.41	1.88	3.49	0.12
6	2.93	1.81	3.02	4.49		3.32	0.41
7	3.26	1.78	3.12			3.59	0.07
Reference	3.43	1.82	3.12	3.24	1.84	3.53	0
Deviation	0.45	0.04	0.13	0.43	0.08	0.12	0.21

Predicted mean response time for the $UP1$ of Media Store with SPE							
$p \in P^{MS}_{SPE}$	v^{MS}_0	v^{MS}_1	v^{MS}_2	v^{MS}_3	v^{MS}_4	v^{MS}_5	Mean deviation
9	1.34	1.18	1.92	1.53		1.46	0.13
10	1.34	1.17	1.32	1.51	1.15	1.47	0.02
11	0.95	0.79	0.95	1.12	0.81	0.98	0.37
12	1.34	1.17	1.33	1.50	1.15	1.47	0.02
13	1.34	1.17	1.33	1.27		1.47	0.05
14	0.95	0.79	0.95	1.12	0.81	0.98	0.37
15	1.42	1.26	1.40	1.58	1.23	1.56	0.10
16	1.21	1.05	1.20	1.21	1.03	1.24	0.15
17	1.34	1.17	1.33	1.34	1.29	1.47	0.07
Reference	1.33	1.17	1.32	1.50	1.05	1.46	0
Deviation	0.11	0.11	0.17	0.17	0.16	0.15	0.15

Predicted mean response time for the $UP2$ of Media Store with SPE							
$p \in P^{MS}_{SPE}$	v^{MS}_0	v^{MS}_1	v^{MS}_2	v^{MS}_3	v^{MS}_4	v^{MS}_5	Mean deviation
9	9.45	4.07	9.40	3.18		10.24	1.34
10	9.46	4.05	9.41	9.66	4.22	10.24	0.29
11	8.42	3.01	8.42	8.63	3.45	8.46	0.82
12	9.46	4.06	9.41	9.61	4.30	10.24	0.30
13	9.46	4.06	9.26	8.69		10.24	0.17
14	8.42	3.01	8.42	8.63	5.45	8.46	0.94
15	9.90	4.59	9.81	3.66	4.66	11.18	1.51
16	8.50	3.50	8.46	8.47	3.83	8.57	0.66
17	9.46	4.06	9.41	5.68	4.54	10.24	0.77
Reference	9.46	3.98	8.93	8.95	4.09	9.88	0
Deviation	0.39	0.38	0.51	1.90	0.51	0.81	0.75

Predicted mean response time for the $UP1$ of Web Server with Palladio							
$p \in P^{WS}_{Pal}$	v^{WS}_0	v^{WS}_1	v^{WS}_2	v^{WS}_3	v^{WS}_4	v^{WS}_5	Mean deviation
10	0.80	0.64	1.13	0.79	0.47	0.76	0.02
11	0.82	0.66	1.15	0.78			0.03
12	0.82	0.66	1.10	0.80	0.47	0.78	0.01
13	0.81	0.65	1.11	0.84	0.47	0.77	0.01
14	0.82	0.66	1.11	0.79	0.58	0.78	0.03
15	0.79	0.50	1.11				0.05
16	0.82	0.30	1.13	0.83	0.47	0.83	0.08
17	0.81	0.66	1.13	0.80	0.48	0.43	0.08
Reference	0.81	0.64	1.10	0.84	0.48	0.78	0
Deviation	0.01	0.07	0.02	0.04	0.02	0.07	0.04

Predicted mean response time for the $UP2$ of Web Server with Palladio							
$p \in P_{Pal}^{WS}$	v_0^{WS}	v_1^{WS}	v_2^{WS}	v_3^{WS}	v_4^{WS}	v_5^{WS}	Mean deviation
10	1.31	0.64	3.48	1.44	0.44	1.38	0.20
11	1.38	0.94	4.32	1.35			0.26
12	1.48	0.94	3.43	1.46	0.45	1.37	0.16
13	1.58	0.87	4.02	1.70	0.44	1.38	0.06
14	1.37	0.79	4.78	1.29	0.45	1.34	0.26
15	1.25	0.53	3.35				0.44
16	1.46	0.29	3.80	1.52	0.47	1.42	0.17
17	1.54	0.88	4.36	1.36	0.43	1.17	0.17
Reference	1.69	0.84	3.93	1.58	0.44	1.38	0
Deviation	0.27	0.17	0.43	0.17	0.01	0.05	0.18

Predicted mean response time for the $UP1$ of Web Server with SPE							
$p \in P_{SPE}^{WS}$	v_0^{WS}	v_1^{WS}	v_2^{WS}	v_3^{WS}	v_4^{WS}	v_5^{WS}	Mean deviation
1	0.76	0.60	1.00	0.73	0.44	0.72	0.02
2	0.76	0.61	1.08	0.74	0.44	0.72	0.01
3	0.76	0.61	1.05	0.77	0.43	0.73	0.00
4	0.76	0.61	1.10	0.77	0.40	0.76	0.02
5	0.76	0.61	1.10	0.77	0.55	0.76	0.03
6	0.77	0.65	1.04	0.77	0.44	0.73	0.01
7	0.76	0.61	1.04	0.81	0.42	0.72	0.01
8	0.83	0.62	1.12	0.77	0.44	0.73	0.03
Reference	0.76	0.61	1.04	0.77	0.44	0.72	0
Deviation	0.00	0.01	0.03	0.02	0.03	0.01	0.02

Predicted mean response time for the $UP2$ of Web Server with SPE							
$p \in P_{SPE}^{WS}$	v_0^{WS}	v_1^{WS}	v_2^{WS}	v_3^{WS}	v_4^{WS}	v_5^{WS}	Mean deviation
1	1.39	0.86	3.70	1.37	0.58	1.26	0.05
2	1.39	0.86	4.16	1.38	0.58	1.27	0.12
3	1.40	0.87	3.76	1.53	1.08	1.27	0.12
4	1.31	0.82	4.24	1.42	0.44	1.31	0.13
5	1.39	0.85	4.85	1.45	0.52	1.39	0.24
6	1.39	0.87	3.74	1.51	0.32	1.29	0.03
7	1.39	0.86	3.73	1.52	0.47	1.26	0.01
8	1.45	0.89	4.82	1.34	0.59	1.30	0.26
Reference	1.39	0.87	3.72	1.52	0.44	1.27	0
Deviation	0.01	0.01	0.31	0.07	0.17	0.03	0.1

C.2 Duration and Break Down

$p \in P_{Pal,MS}$		1	2	3	4	5	6	7
Duration reading		40	35	25	59	50	38	45
Control flow	modelling	30	23		60	25	23	
Resource demand	modelling	15	17		20	38	9	
Resource environment	modelling	5	15		10	3	24	
UP1	modelling	10	26		15	28	16	
	searching for errors	25	15			35	0	
	analysis	10	24			1	34	
UP2	modelling	10	25			4	6	
	searching for errors	25	0			112	0	
	analysis	5	7			1	2	
Duration v0		135	152	135		247	114	
v1	modelling	25	26	35		16	33	
	searching for errors	5	32	5		16		
	analysis	5	58	5		1	32	
Duration v1		35	116	45		33	65	
v2	modelling	25		30		7	23	
	searching for errors	5		5		7		
	analysis	10		10		1	13	
Duration v2		40		45		15	36	
v3	modelling	5		10		8	2	
	searching for errors			0		5	3	
	analysis	10		5		1	2	
Duration v3		15		15		14	7	
v4	modelling	60		45		9		
	searching for errors	0		0		37		
	analysis	0		15		1		
Duration v4		60		60		47	0	
v5	modelling	0		10		1	4	
	searching for errors	0		0		3		
	analysis	5		10		1	3	
Duration v5		5		20		5	7	
Rank variants			2	10		16	2	
Additional question				10			1	
Turn in				0			8	
Duration for questionnaires		0	2	20	0	16	11	0
Duration for work on task		290	270	340	?	377	240	345
Overall duration (minutes)		330	305	365	?	427	278	390

$p \in P_{SPE,MS}$		9	10	11	12	13	14	15	16	17
Duration reading		22	25	20	38	20	24	27	30	45
Control flow	modelling		20	20	20	55	18	19	8	10
Resource demand	modelling									
Resource environment	modelling		7	20	16	15	18	16	20	12
UP1	modelling	74	9	15	13	20	4	12	0	18
	searching for errors		9						4	
	analyse		12		3	10	20	6		5
UP2	modelling	21	3	25	3	20	2	7	22	5
	searching for errors		2							
	analyse		1		2	5	1	10		5
Duration v0		95	63	80	57	125	63	70	54	55
v1	modelling		15	10	9	28	13	15	39	19
	UP2		4	10	3	5	3	5	15	3
	searching for errors		0				1			
	analyse		0				8	10	1	
Duration v1		25	19	20	12	33	25	30	55	22
v2	modelling		10	20	39	40	10	20	31	15
	UP2			23	8		30	5	14	3
	searching for errors					13				1
	analyse		24		4			5		
Duration v2		?	34	43	51	53	40	30	45	19
v3	modelling		17	30	35	65	54	75	38	15
	UP2		15	15	9			5	5	9
	searching for errors		30		10	25				10
	analyse				3	13		5		
Duration v3		?	62	45	57	103	54	85	43	34
v4	modelling		25	50	30		12	30	21	30
	UP2		15	20	25		63	15	10	2
	searching for errors								21	
	analyse				3			5		8
Duration v4		?	40	70	58	0	75	50	52	40
v5	modelling		10	10	9	17	18	20	4	11
	UP2		7	10	1			5	12	4
	searching for errors									
	analyse							5		2
Duration v5		?	17	20	10	17	18	30	16	17
Rank variants			8	2	10		1		7	8
Additional question			10				1	28		20
Turn in			20							
Duration for questionnaires		0	38	2	10	0	2	28	7	28
Duration for work on task		349	273	280	255	331	277	323	272	215
Overall duration (minutes)		371	298	300	293	351	301	350	302	260

$p \in P_{Pal,WS}$		10	11	12	13	14	15	16	17
Duration reading		20	40	21	20	30	25	14	20
Estimation of ranking		6	5	4	5	25	5	4	3
Control flow	modelling	22	65	54	40	54	50	71	77
Resource demand	modelling	12	60	24	30	0	30	0	10
Resource environment	modelling	4	5	13	20	29	10	5	10
UP1	modelling	11	22	12	22	10	30	8	30
	searching for errors	10	95	11	3	15	100	47	10
	analyse	10		12	0	5	10	0	0
UP2	modelling	29	10	10	5	3	20	42	5
	searching for errors	0	0	14	0	0	0	0	0
	analyse	0	0	10	5	5	10	0	0
Duration v0		98	257	160	125	121	260	173	142
v1	modelling	16	10	24	15	27	30	28	15
	searching for errors							17	10
	analysis UP1	7		2	5	2	10		
	analysis UP2			2	5	2	10		
Duration v1		23	10	28	25	31	50	45	25
v2	modelling	1	15	3	2	5	10	2	2
	searching for errors								
	analysis UP1	4		1	2	1	5		
	analysis UP2			2	3	1	5		
Duration v2		5	15	6	7	7	20	2	2
v3	modelling	16	40	17	25	15	15	27	15
	searching for errors			6			10	23	
	analysis UP1	2		2	5	5			
	analysis UP2	5		1	5	5			
Duration v3		23	40	26	35	25	25	50	15
v4	modelling	8		10	25	5		20	10
	searching for errors							3	15
	analysis UP1	1		2	5	1			
	analysis UP2	7		2	5	2			
Duration v4		16	0	14	35	8	0	23	25
v5	modelling	7		10	15	20		10	15
	searching for errors							5	
	analysis UP1	1		3	3	2			
	analysis UP2	2		2	2	3			
Duration v5		10	0	15	20	25	0	15	15
Duration all variants		77	65	89	122	96	95	135	82
Average		15	22	18	24	19	32	27	16
Ranking		44			68	30		10	10
Turn in Duration questionnaires		44	0	0	68	30	0	10	10
Duration for work on task		219	322	249	315	247	355	318	234
Overall duration (in minutes)		239	362	270	340	277	380	332	254

$p \in P_{SPE,WS}$		1	2	3	4	5	6	7	8
Duration reading		20	27	20	24	20	40	30	31
Estimation of ranking		5	4	5	2	3	20	30	7
Control flow	modelling	20	35	30	67	15	10	85	86
Resource environment	modelling	10	15	10		23	5		15
UP1	modelling		25		99	11	5		
	searching for errors		4	5		8	5		
	analyse	5	2	2		2	5	1	
UP2	modelling		16			17	4	10	12
	searching for errors		5			37			
	analyse	5	2	2		2	20	1	
Duration v0		40	104	49	166	115	54	97	113
v1	modelling	10	18	5	6	5	10	15	20
	UP2	5	2	5		3	3	4	
	searching for errors		8			3			
	analyse	5	2	5		3	4	1	
Duration v1		20	30	15	6	14	17	20	20
v2	modelling	5	14	5	5	3	10	5	17
	UP2	5	4	1	5	4	10	3	5
	searching for errors		3			2			
	analyse	5	2	5		1	5	1	
Duration v2		15	23	11	10	10	25	9	22
v3	modelling	5	11	8	15	16	4	15	15
	UP2	5	7	5	2	2	3	5	10
	searching for errors		3			1		30	
	analyse	5	2	5		3	6	5	
Duration v3		15	23	18	17	22	13	55	25
v4	modelling	35	22	35	22	16	15	20	40
	UP2	15	8	10		3	10	10	6
	searching for errors		12	5		3	10	25	
	analyse	5	2			3	10	5	
Duration v4		55	44	50	22	25	45	60	46
v5	modelling	25	3	5	15	2	2	5	7
	UP2	15	18	35		32	3	25	31
	searching for errors		3			1		5	
	analyse	5	2	5		1	10	1	
Duration v5		45	26	45	15	36	15	36	38
Duration all variants		150	146	139	70	107	115	180	151
Average		30	29	28	14	21	23	36	30
Ranking variants		5	3	20		11	15	4	
Turn in Duration questionnaires		5	3	20	0	11	15	4	0
Duration for work on task		195	253	208	236	233	184	281	264
Overall duration (in minutes)		215	280	228	260	253	224	311	295

C.3 Problems

In the section C.3.1, the actual recorded problems are presented. The first 4 tables present the question protocol for both the Media Store task and the Web Server task. The following 4 tables present the collected problems from the acceptance test for both tasks. Finally, 4 tables present the problems found in the final models.

In the section C.3.2, the amount of problems for each category is presented, still separated by the three dimensions occurence, problem area and severity.

C.3.1 Record of Problems

In the in line tables, the following keys for problem area and problem severity are used due to space limitations:

Area: 0 = Tool, 1 = Methodology, 2 = Task

Severity: 2 = major, 1 = intermediate, 0 = minor

Question Protocol

Question protocol Media Store and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Komponentenparameter allgemein	1	Component parameters	2
1	Wo Komponentenparameter spezifizieren?	0	Usage	2
1	Mittelwert als Komponentenparameter verwendet - > nur bei SPE-ED verwenden!	2	Usage profile	1
1	Komponentenparameter an zuviele Komponenten gehängt	1	Component parameters	1
1	Wie Name des Komponentenparameters bestimmen? -> einfach angeben	0	Usage	1
1	Komponentenparameter wird nicht in StoEx angezeigt -> geht trotzdem	0	Usage	1
1	Neue Verteilung bei Encoding auch dort als Komponentenparameter? -> Nein, parametrisieren	1	Component parameters	1
1	Charakterisierung des Komponentenparameters mit angeben	1	Component parameters	2
4	Return in zwei SetVariableActions	1	Parameters	1
1	Wie wird RETURN charakterisiert? SetVariableAction	1	Parameters	2
1	Rückgabewerte der DB modellieren, Größe der Files	1	Parameters	1

Question protocol Media Store and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Andere Parametrisierung verwendet als gesetzt: Hinweis: Genau die setzen die gebraucht, STRUCTURE ist kein Oberbegriff für alle	1	Parameters	2
1	INNER Charakterisierung einer Collection unklar	1	Parameters	2
1	Outparameter falsch zugewiesen	1	Parameters	1
1	EA1 anstatt NoE = 1 im Cache NoE = files.NoE -> too many users spawned	1	Parameters	2
1	Wusste nicht mehr, dass es EnumPDF gibt	1	Parameters	1
2	Konvertierung DoublePDF in Integer (Trunc)	1	Types and units	1
1	Umrechnungsproblem MB <-> Byte	1	Types and units	1
1	Erklärt das Bytesize nicht explizit mal NoE genommen werden muss	1	Types and units	1
1	Wie BP2 modellieren, neues Scenario oder neues File? -> neues File	0	Usage	1
1	Hinweis: Alte Simulation verwendet bzw. falsches Eclipse	0	Bug	1
1	Unvollständiger Usage Flow	0	Error	1
1	Repository der Primitive Types ausgewählt	0	Error	1
1	Entwurf auf falsches Resource Environment allokiert. Dadurch Demand -> unendlich	0	Error	1
1	Wie ist Verteilung der Anzahl der Dateien gemeint?	2	Usage Profile	1
1	Verteilung Dateigrößen falsch verstanden	2	Usage Profile	1
1	Loop fehlte in SEFF, da nicht in Sequenzdiagramm deutlich (nur im Text)	2	Control flow	2

Question protocol Media Store and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Encode mit 4 Branches	2	Control flow	1
1	Anwendungsfälle alle in ein Szenario? Nur nochmal bestätigt			
1	Verteilung: Im DB Szenario Werte aus Branches nehmen	1	Distributed system	2
1	Studi merkte selbst dass zu hohe Werte in EA3 -> Ankunftsrate DB Szenario			
1	System bei Verteilung... (Frage vergessen)	1	Distributed system	1

Question protocol Media Store and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Warum nicht DB auf mehrere Szenarien: Weil auf einer Maschine	1	Distributed system	1
1	Wie kombiniert man die beiden Szenarien zu einem?	1	Distributed system	2
2	Wie Processing Rate angeben, wie weit runden	1	Overhead	0
1	Wie SPE Projekt vervielfältigen	1	Projects	0
1	Berechnung der Wartezeit am Pool: Wie weiter, wenn Zeit zwischen Benutzern bekannt? Hinweis: Wie lange braucht Benutzer den Pool	1	Passive Resource	0
1	Berechnung Pool: Kann man ganz klein auf 0 runden? ja	1	Passive Resource	0
3	Berechnung Pool: Korrigiert das HD Zugriff nicht in Pool mit drin	1	Passive Resource	0
1	Netzwerk zu System Entry wird nicht modelliert?	2	Control flow	1
1	SPE: Zugriff auf Cache kostet nichts?	2	Annotation	0
1	SPE: Müssen Verteilungen in Aufgabenstellung berücksichtigt werden?	2	Usage Profile	0

Question protocol Web Server and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
2	Wie spezifiziert man Komponentenparameter -> im System	0	Usage	2
1	Wie spezifiziert man Komponentenparameter im System -> Baum, Rechtsklick... gezeigt	0	Usage	2
1	Größe der Dateien wirklich als Komponentenparameter?	1	Component parameters	0
1	Wie spezifiziert man den Namen einer Komponentenparameters?	0	Usage	1
1	Wie modelliert man die zwei Charakterisierungen im UM? Hinweis: EnumPMF	1	Parameters	1
1	Out of Memory: Internal Demand falsch: nicht durch 1000 geteilt.	1	Types and units	2
1	Wo ist das System Diagram zu finden -> Composite benutzen	0	Usage	2
1	RequestAnalyser nicht im Repository gefunden	0	Usage	0
1	Wie Usage Model anlegen?	0	Usage	2
2	Wie etwas in ForkedBehaviour angeben? -> Resource Demanding Behaviour hineinziehen	0	Usage	2
1	Compilerfehler -> Rollen falsch verbunden	0	Error	1
2	Compilerfehler -> Leerer StoEx	0	Error	1

Question protocol Web Server and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Hatte Usagemodell und Diagramm kopiert und damit Modell kaputt gemacht.	0	Usage	1
1	Beim Anlegen eines AssembyConnectors hat der Editor einen ChildCompoContext auf null gesetzt. Unklar	0	Bug	1
1	Frage zu Resource Demands: Egal ob 1WU.(0 oder 1 HD) oder (1 WU oder 1WU+1HD)?	0	Usage	0
1	Probleme beim Anlegen von Forks, Tool Restart half	0	Bug	0
1	Exception: BranchAction mit nur einer Guarded-BranchTransition.	0	Error	1
2	Wo Broker einstellen?	0	Usage	2
1	EA1: Modelle referenzierten sich falsch	0	Bug	2
1	Wird Name für Usage Model vergeben, ändert sich die Ausgabe bei den Sim-Ergebnissen, ”Usage Model” als String verschwindet.	0	Bug	0
1	Unklar, dass in Allokation keine Linking Resource dargestellt.	0	Usage	1
1	Validierer mag keine Forks ohne Startzustand	0	Error	1
1	Fehler in Generatorlauf -> Falsche Allokation	0	Error	1
1	Im Usage Model Iterations auf 0	0	Usage	1
1	Out of Memory: Resource Environment falsch: Disc Processing Rate falscch bestimmt	1	Types and units	2
1	Haben statische Seiten auch MM Inhalte?	2	Control flow	1
1	Richtig dass keine Kosten für den Aufruf der Komponente?	2	Annotation	1

Question protocol Web Server and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Wo globale Parameter angeben?	0	Usage	1
1	Wie Wahrscheinlichkeiten angeben? -> Hinweis auf Wahrscheinlichkeitsbaum und VL	1	Probabilities	2
1	Wie Tool klarmachen, dass einmal HTML und mal MM abgefragt wird? Möglichkeiten erläutert (Nur Mittelwerte oder UM mit modellieren)	1	Probabilities	2
1	Wie ist die Wahrscheinlichkeit für HTML und MM -> gesagt das 1 / (1+2.5) bzw 2.5/(1+2.5)	1	Probabilities	2
1	Wie p(statisch)ermitteln, was danach	1	Probabilities	2
1	Frage ob man auch direkt UM modellieren kann ohen Bayes -> ja klar	1	Probabilities	0

Question protocol Web Server and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Frage ob für die Performanz nicht relevantes weggelassen werden kann -> im Prinzip ja (macht er aber nun doch nicht.			
1	Abschätzung auf 0 ok? -> ja	1	Passive Resource	0
2	Wie Abschätzen p(Verweilzeit > x)? Einfach schätzen	1	Passive Resource	0
1	Wie abschätzen? Hinweise geben.	1	Passive Resource	0
1	Rechnung so richtig? -> noch hoch 8, weil für 8 Benutzer hintereinander	1	Passive Resource	0
1	Bei EA5 BP1 nichts verändert: Doch, 10 WU weg	2	Annotation	1
1	StaticFileProvider kein WU um zu prüfen welcher Request-Typ?	2	Annotation	0
1	Wird pro Benutzer oder pro Request ein Thread angefordert?	2	Control flow	1

Problems in Acceptance Tests

Acceptance test Media Store and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA1 BP2: NoE im Cache war erst nicht 1. Weiterhin verdächtiger Peak bei 0.	2	Control flow	2

Acceptance test Media Store and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA3 BP1: 1x Fehler in Overhead (nicht aktualisiert für DB, zweite Disk)	1	Overhead	2
1	EA3 BP1: 1x Netzwerk vergessen	1	Overhead	0
1	EA3 BP1: Service Times des DBServers wurden nicht angepasst.	1	Overhead	2
1	EA3 BP2: Im Overhead fehlte 1 für Delay. Danach Ergebnis komisch, aber keine Gründe zu finden.	1	Overhead	2
1	EA0 BP1: Im Overhead fehlte 1. EA2 BP2: Delay verändert.	1	Overhead	2
1	EA1 BP1: Cache nur einmal abgefragt, nicht pro File. Hat Ergebnisse mit Sitznachbar verglichen.	2	Control flow	2
1	EA4 BP1: variable neu kodiert. Neukodierung vergessen.	2	Control flow	1

Acceptance test Media Store and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA0 BP1: Watermark fehlerhaft auf HDD anstatt CPU. EA3 BP1: Werte aus dem Inneren der Branches nehmen	1	Overhead	2
1	EA4 BP1: 4. Fall konstant, <192 fehlte.	2	Control flow	1
1	EA3 BP2: DB Sachen in ein Szenario oder Service Level anpassen. Danach zwar komisches Ergebnis, aber keine Gründe gefunden.	1	Distributed System	2
1	EA4 BP1: 4. Fall nicht berücksichtigt(variabel, >192)	2	Control flow	1

Acceptance test Web Server and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA0: EA0: Korrektur von static/dynamic bei MM Daten.	2	Control flow	2
1	VALUE anstatt BYTESIZE bei Kompparams verwendet.	1	component parameters	1
1	EA3: Versehentlich zuerst AccContentGenerator verwendet.	1	Assembly	2
1	EA0: Eine Parameterübergabe war fehlerhaft	1	Parameters	1
1	EA0 BP2: request.type und request.url.type vertauscht.	1	Parameters	1
1	EA0: Komponentenparameter als IntPMF modelliert, wusste nicht, wie DoublePDF geht.	1	Types and units	1
1	EA4: Return Value nicht modelliert.	1	Parameters	1
1	EA0: Verteilungen im SEFF anstatt KomParams, DoublePMF anstatt DoublePDF	1	Types and units	1
1	EA0: Usage Model umständlich und fehlerhaft.	1	Usage model	2
1	EA1: Modelle referenzierten sich untereinander fehlerhaft	0	Bug	2

Acceptance test Web Server and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA0: Hatte für jedes KB 1 HD Access.	1	Overhead	1
1	EA4: Bei dynamisch nicht 50 von HD laden: Macht aber kaum etwas aus.	2	Annotation	0
1	EA0 : Wahrscheinlichkeiten: nur % stat/dyn betrachtet, nicht MM.	1	Probabilities	1
1	EA1: p(CacheHit) und p(CacheMiss) verwechselt.	2	Control flow	2

Acceptance test Web Server and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	EA2: Falsch kopiert von Cache.	0	Usage	2
1	EA3: Demand zum Erzeugen eines Threads nicht dabei.	2	Annotation	1
1	EA4: Delay war auf 0.	1	Overhead	2
1	EA0 BP2: Selbst zu hohe Zeit erkannt. Hinweis: in Overhead auch HD angeben.	1	Overhead	1
1	Wahrscheinlichkeiten p(HTM|stat) usw. berechnen.	1	Probabilities	1
1	EA4: Bei Aufruf des Appservers nochmal zwischen statisch / dynamisch unterschieden, daher zu niedrige Werte	1	Distributed system	1
1	EA0: Zu niedrig, war nur Antwortzeit für einen Request: mal 3,5 nehmen.	1	Projects	2
1	EA4: Problem mit Service Time im Overhead.	1	Overhead	1
1	EA4 BP2: War No Contention Lösung, daher erst zu niedrig	1	Solution	1
1	EA3: Fehler in Overhead, 1 WU = ...	1	Overhead	1
1	EA4: Overhead Appserver CPU nicht doppelt so schnell gemacht. Loop war zu oft, selbst gefunden.	2	Annotation	2
1	EA2:Mal 1 MB von HD gelesen, auch schon in EA0, aber nicht korrigiert	1	Overhead	1

Problems in Final Models

Final models Media Store and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Filesize auch bei Upload als Komponentenparameter	1	Component Parameters	0
2	Filesize alsKoParam der WebGUI	1	Component Parameters	0
2	Konstant und >192 nicht berücksichtigt	2	Control flow	1
1	Modelle referenzieren sich falsch, daher Cache nicht benutzt	0	Bug	2
1	Dateigröße in ganzen MB angegeben als Double angegeben. Macht aber nichts, da entspr. *1E6 genommen. Nur leichte Abweichungen da manchmal nicht auf Byte gerundet	1	Types and units	0

Final models Media Store and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Only 1 WU instead of 3 to get Filename from the database(EA0 UP1, UP2 is right)	2	Annotation	0
2	0.00019 for CPU service time instead of 0.001	1	Overhead	1
1	modelled lan as delay for EA3 UP1	1	Overhead	0
1	EA3 UP1 Overhead defekt, delay für DB Server weg (kein Fehler von Henß)	0	Bug	0
1	Added 1 WU for each component call	2	Annotation	0
1	Service Time CPU 0.001145 for 1 WU is too long	1	Overhead	1
2	DB Server has two hard disks, thus results are much lower.	1	Overhead	2
1	BP2 used 7.1 as mean for filesize, not 8.1	2	Usage Profile	1
1	EA4 UP1 3.5*1.33*10*0.44 is not representing the encoding	1	Miscalculation	1
1	EA3 used time of branch node, which is multiplied with probability and thus too low.	1	Distributed system	1
1	Tiny error: WU to read from DB using the DB Connection Pool is 0.2 too high	2	Annotation	0
1	Used Latency instead of transfer time for LAN, which is lower by factor 50	1	Overhead	1
1	slightly wrong encoding: 28.292 instead of 25.102 WU	1	Miscalculation	0
1	EA5 10 WU instead of 17 for very first component lookup	2	Annotation	0
3	EA3: Did not model the network	1	Overhead	1
1	EA4: Used commas instead of dots for encoding demand: encoding by factor 10 slower!	0	Usage	1
1	EA0 UP2: Didn't adjust WUs for reading from DB for number of files, 3 WU difference	2	Annotation	1
1	EA1: Reading from DB not after cache checking	2	Control flow	0
1	EA3 UP1: Used time to download from DB server for delay for upload as well	1	Distributed system	1

Final models Web Server and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Beim Threadpool trotzdem 10 WU zum Erzeugen	2	Annotation	1
3	Demand zum erzeugen des 2. Thread bei paralleled logging vergessen	2	Annotation	1
1	Demand zum Parsen der Seite bei FastRequest-Analyser (ParLog) vergessen	2	Annotation	1

Final models Web Server and Palladio				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	Keine Komponentenparameter verwendet, sondern Verteilungen direkt ins Modell	1	Component parameters	0
1	Trunc vor *1000 bei Dateigröße, also immer auf KB gerundet.	1	Types and units	1
1	DynamicFileProvider does not return a filesize (thus, it cannot be used for EA4)	1	Parameters	1
1	DoublePMF instead of DoublePDF for the filesizes, strongly influences the distributions and the deviation (was detected after looking at the results)	1	Types and units	2

Final models Web Server and SPE				
Quantity	Question / Problem / Answer	Area	Class	Severity
1	paralleled logging anders modelliert, in 50% der Fälle 2. Logging, immer 1. Logging	2	Control flow	1
3	Beim Threadpool trotzdem 10 WU zum Erzeugen	2	Annotation	1
5	HD Latenz belastet Delay, nicht Festplatte	1	Overhead	1
3	EA4: Analysed the DB scenarios separately for UP2, not in one for contention	1	Distributed system	1
1	Demand for dynamic MM generation slightly wrong	1	Miscalculation	0
1	Lookups are all outside the branches, but two should be in dynamic branch	2	Control flow	1
1	left out response generation instead of logging for EA3	2	Annotation	0
1	Latency of 1 ms for LAN in EA4	1	Overhead	1
1	Added HD Access for dynamic MM	2	Annotation	1
1	put lookup for static file provider in branch, but it is always needed (Chain of Resp)	2	Control flow	1
1	Forgot demand to create a second thread in paralleled logging alternative	2	Annotation	1
1	Latency LAN is 5ms instead of 0.5ms	1	Overhead	1
1	Latency of LAN is added for each transfered KB	1	Overhead	1
1	DB Server has arrival rate of 3.5 and full probabilities in the branches, i.e. load is too high	1	Distributed system	1
2	EA4: Did not model the network	1	Overhead	1
1	EA4: Service time of called appserver is too high.	1	Distributed system	1
1	EA4: Interchanged (mixed up) probabilities for HTML and Multimedia in appserver	1	Probabilities	1
1	EA3: Twice a new second thread is generated, thus 10 WU too much per request	2	Annotation	1

Severity	Control flow	Annotation	Usage Profile
Media Store, 16 participants			
minor	0	1	0
intermediate	2	0	3
major	1	0	0
Web Server, 16 participants			
minor	0	1	0
intermediate	0	2	1
major	0	0	0

Table C.13: M2.1 Number of questions concerning the experiment task

C.3.2 Cumulated Data

The first three tables show the number of question from the question protocol: Related to the task (table C.13, related to Palladio (table C.15) and related to SPE (table C.17).

Following, the next three tables show the number of problems in the acceptance test: Related to the task (table C.18, related to Palladio (table C.20) and related to SPE (table C.22).

Finally, the last three tables show the number of problems in the final models: Related to the task (table C.23, related to Palladio (table C.25) and related to SPE (table C.27).

C.4 Answers to Questionnaire

C.4.1 Question 17

Answers to question 17: Improvements for the Palladio approach and the PCM Bench:

> Automatische Weitergabe der Parametercharakterisierungen; Bessere Fehlererkennung; Parametrisierung; Verbesserung der Analysemöglichkeiten (bessere Vergleichsdiagramme); Verbesserung der Editoren (textuelle Teilmodellierung); flexiblere graphische Editoren (Copy and Paste, Drag & Drop); Ein Click-through-wizard für alle elementaren Elemente; Automatische Code-Generierung; Verstecken der nicht benötigten Schaltflächen; Viele Bugs; Variablen durchreichen unübersichtlich; Oft nicht klar wann keine Autovervollständigung und wann fehlerhafte Meldung; Möglicherweise wäre es sinnvoll beim Spezifizieren eines Komponentenaufrufs darüber informiert zu werden, welche Parameter spez. werden müssen; Bessere Trennung der Rollen, z.B. durch verschiedene Sichten; EMF bedingte Schwachstellen, die bereits im Bugzilla stehen; Skript-Anbindung für den Kontrollfluss in SEFF; Kopierfunktion zwischen den SEFF; Textuelle konkrete Syntax: Alternatives UI für Entwickler mit Hang zum Code (und zur Effizienz); Bugfixing fürs Werkzeug; Doku; GUI-Konzept auf Schnelligkeit auslegen; Algorithmische Bewertung neben den SEFFs; Die Handhabung der GUI: Erzeugen aller Modelle/Diagramme über ein „new"; Einheitliche Erstellungsmethode für die Modelle und Diagramme; Oft unklar, „wo man hinklicken soll" in der grafischen Darstellung; Usage Model(?) wieso open/closed belastung „außerhalb des Kastens"; Bugs fixen; mehr graphische Editoren statt Baumansicht; verschachtelte SEFFs -> übersichtlicher; plattformunabhängigkeit

C.4.2 Question 21

	Tool			Methodology				
	Usage	Error	Bug	Parameters	Component parameters	Types and units	Assembly	Usage model
Media Store, 7 participants								
minor	0	0	0	0	0	0	0	0
intermediate	3	3	1	7	2	4	0	0
major	1	0	0	4	2	0	0	0
Web Server, 8 participants								
minor	2	0	2	0	1	0	0	0
intermediate	7	3	1	1	0	0	0	0
major	9	0	1	0	0	2	0	0

Table C.15: M2.1 Number of questions concerning Palladio

	Tool			Methodology						
	Usage	Error	Bug	Overhead	Distributed system	Projects	Passive Resource	Miscalculation	Probabilities	Solution
Media Store, 9 participants										
minor	0	0	0	2	0	1	5	0	0	0
intermediate	0	0	0	0	2	0	0	0	0	0
major	0	0	0	0	2	0	0	0	0	0
Web Server, 8 participants										
minor	0	0	0	0	0	0	5	0	1	0
intermediate	1	0	0	0	0	0	0	0	0	0
major	0	0	0	0	0	0	0	0	4	0

Table C.17: M2.1 Number of questions concerning SPE

	Control flow	Annotation	Usage Profile
Media Store, 16 participants			
minor	0	0	0
intermediate	3	0	0
major	2	0	0
Web Server, 16 participants			
minor	0	1	0
intermediate	0	1	0
major	2	1	0

Table C.18: M2.1 Number of problems in the acceptance test concerning the experiment task

	Tool			Methodology				
	Usage	Error	Bug	Parameters	Component parameters	Types and units	Assembly	Usage model
Media Store, 7 participants								
minor	0	0	0	0	0	0	0	0
intermediate	0	0	0	0	0	0	0	0
major	0	0	0	0	0	0	0	0
Web Server, 8 participants								
minor	0	0	0	0	0	0	0	0
intermediate	0	0	0	3	1	2	0	0
major	0	0	1	0	0	0	1	1

Table C.20: M2.1 Number of problems in the acceptance test concerning Palladio

	Tool			Methodology						
	Usage	Error	Bug	Overhead	Distributed system	Projects	Passive Resource	Miscalculation	Probabilities	Solution
Media Store, 9 participants										
minor	0	0	0	1	0	0	0	0	0	0
intermediate	0	0	0	0	0	0	0	0	0	0
major	0	0	0	5	1	0	0	0	0	0
Web Server, 8 participants										
minor	0	0	0	0	0	0	0	0	0	0
intermediate	0	0	0	5	1	0	0	0	2	1
major	1	0	0	1	0	1	0	0	0	0

Table C.22: M2.1 Number of problems in the acceptance test concerning SPE

Severity	Control flow	Annotation	Usage Profile
Media Store, 16 participants			
minor	1	4	0
intermediate	2	1	1
major	0	0	0
Web Server, 16 participants			
minor	0	1	0
intermediate	3	11	0
major	0	0	0

Table C.23: M2.1 Number of errors in the model concerning the experiment task

	Tool			Methodology				
	Usage	Error	Bug	Parameters	Component parameters	Types and units	Assembly	Usage model
Media Store, 7 participants								
minor	0	0	0	0	3	1	0	0
intermediate	0	0	0	0	0	0	0	0
major	0	0	1	0	0	0	0	0
Web Server, 8 participants								
minor	0	0	0	0	1	0	0	0
intermediate	0	0	0	1	0	1	0	0
major	0	0	0	0	0	0	0	0

Table C.25: M2.1 Number of errors in the model concerning Palladio

	Tool			Methodology						
	Usage	Error	Bug	Overhead	Distributed system	Projects	Passive Resource	Miscalculation	Probabilities	Solution
Media Store, 9 participants										
minor	0	0	1	1	0	0	0	1	0	0
intermediate	1	0	0	7	2	0	0	1	0	0
major	0	0	0	2	0	0	0	0	0	0
Web Server, 8 participants										
minor	0	0	0	0	0	0	0	1	0	0
intermediate	0	0	0	10	5	0	0	0	1	0
major	0	0	0	0	0	0	0	0	0	0

Table C.27: M2.1 Number of errors in the model concerning SPE

Bessere Abbildung für verteilte Systeme; Schwer zu bedienen; Da weniger Einflüsse modellierbar sehr abstrakt (nicht so genau); Benutzerfreundlichkeit des Werkzeugs; Mittelwertbetrachtung; Verwaltung der Szenarien/Konfigurationen; Verschieben etc. im Editor erlauben; Referenzen statt copy-include; Moderne GUI; Diagramme; Umstellung auf moderne Technologie, Tabs, etc.; Neuentwicklung; komplett neu schreiben, insbesondere die GUI; Darstellung; kein wirklicher Aufruf anderer Szenarien, sondern Umweg über Delay; Die Oberfläche, komplett! Mehr auf Modellierung und Wiederverwendbarkeit ausrichten; Bedienerführung (GUI); künstliche Limitierungen entfernen (Feldgröße, max Anzahl Ebenen); Vorgehen näher an das SW-Entwicklungsmodell anlehnen; wegwerfen; Simulationsfunktion unbrauchbar; Probleme mit Branches; Bugs fixen; Benutzeroberfläche modernisieren; Undo, Copy & Paste einbauen; Undo und viele weitere nützliche Funktionen; komplett neu entwickeln

C.4.3 Question 29

Answers to question 29: Reasons for having more trust in the quality of the predictions for one approach:

Weil ich alles genauer modellieren konnte; Da mehr Einflussfaktoren berücksichtigt; bessere Simulation und Betrachtung durch Verteilungsfunktionen; Simulation und Modelle weniger abstrakt; Man kann den Simulationscode einsehen; da Verteilung; wegen der Wahrscheinlichkeitsfunktion; Reduzierung auf Mittel- und Max-Werte bei SPE-ED vielleicht zu einseitig; Komplexität des SW Modells wird hinreichend exakt abgebildet; SPE-ED modelliert ungenau; möglicherweise PCM wegen Verteilungen; Spezifikation der Verteilungen anstatt nur Mittelwerte möglich; Die Vorgänge innerhalb des Tools sind besser nachzuvollziehen, nicht zuletzt, da man auch die Möglichkeit hat, sich das Generat anzusehen; da nicht nur ein Wert, sondern eine Funktion, gerade bei der Simulation und Mehrbenutzerbetrieb; höhere Transparenz; keine (stark) vereinfachten Annahmen (Verteilungen bei SPE); beide haben Probleme, daher wenig Vertrauen; da ich so ungefähr verstehe, wie alles berechnet ist, bei SPE-ED scheint mir der Mittelwert nicht realistisch, außerdem weiß ich nicht genau wie die Performanz bei SPE-ED analytisch bestimmt ist; weil genauer modelliert werden kann; genauer

C.4.4 Question 31

Answers to question 31: Reasons for preferring Palladio:

mehr Realitätsnah; da Bedienung besser war; viel modernerer Ansatz (Komponentenansatz usw.); komplexer, flexibler, mächtiger; Neu-Effekt; Eclipse gewohnte Umgebung; in Weiterentwicklung, Komponentenkonzepte und MDSD Kompatibilität; genauere Trennung der Rollen; genauere Abbildung in Modellen; intuitive Oberfläche; Modellierung kam mir genauer vor; bessere Modellierung; wegen der Bedienung; intuitiver (näher an der SW Entwicklung); SPE-ED ist alle 10 Minuten

abgestürzt; mehr Vertrauen; Usability höher; Arbeit weniger frustrierend, wenn was nicht läuft; interessanter, komfortabler, präziser; nur die vielen Bugs haben genervt; wegen der Benutzeroberfläche

Answers to question 31: Reasons for preferring SPE:

bei PCM nach längerer Zeit gesucht wo Variablen falsch übergeben und nur noch Kästchen gesehen; sehr viel einfacher

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