Carl von Ossietzky University of Oldenburg
Department of Computing Science
Software Engineering Group

Study Thesis (”Individuelles Projekt”)

Empirical Validation and Comparison of the Model-Driven Performance Prediction Techniques of CB-SPE and Palladio

Anne Martens, Mat.-Nr. 8146070

August 12, 2005

Supervised by:
Dipl.-Math. Viktoria Firus
Jun.-Prof. Dr. Ralf H. Reussner

For the design of component based systems, it is important to guarantee non-functional attributes before actually composing a system. Performance usually is a crucial property of a software system, for safety or usability reasons. Several approaches to predict performance characteristics of a component based system in an early stage of development have been introduced in recent times. However, for an engineering discipline, not only the propose of techniques is needed, but also empirical studies of their applicability.

This work empirically compares and evaluates two approaches to early predict the performance of component based software systems. The empirical study is conducted in form of a case study, although attempts are made to achieve a good generalizability.

The results attest the CB-SPE technique a good applicability, although some problems occured. The Palladio technique has less good results. Here, there have been problems with the specification of the distribution functions.

Contents

1 Introduction

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 [11], which leads to high costs or even the failure of the project. Users are not willing to accept long response times, and a high response and processing time disturbs operating the system. The problem here often is not just to guarantee certain performance values for a fix number of users, e.g. some test users, but to guarantee scalability, i.e. guarantee performance values for certain numbers of users. The future load of a system is hard to assess: Often, systems are tested with a number of test users and perform well, but fail to meet performance criteria when used in practice with a much higher user number.

Prominent examples for systems failing and causing high losses because of not complying with performance requirements are the automated baggage handling system at Denver airport and IBM's information system at the Olympic Games 1996 in Atlanta [22]. 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 [17]. The problems with IBM's information system at the Olympics caused the company high losses in reputation, not expressible in numbers [12]. Here, the problem was the number of users, too: The system was tested with 150 users, however, 1000 user accessed it during the Olympic games, causing a system collapse.

In spite of these experiences, the performance of a software system is most 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, [24]). 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.

1.1 Software Performance Engineering

Since the beginning of the 80's, the early analysis of non-functional properties, including performance, has been a topic of research. By analyzing 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 formed by Connie U. Smith in 1981. She later defined it as a "systematic, quantitative approach to constructing software systems that meet performance objectives" [24], with being an "engineering approach to performance, avoiding the extremes of performance-driven development and 'fix-it-later' " [24].

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 [2]. An overview for performance prediction techniques at an architectural level is given in [1].

1.2 Component-Based Software Performance Engineering

The prediction of performance has a critical relevance in the assembling of a system from components. However, in many previous works concerning component-based software engineering, the prediction of functional properties was getting more attention [23].

Component-based software engineering brings potential advantages for the software performance engineering, because the performance of the system arises from the performance of the single components, if their interaction is known. If components are already known with their implementation, they might be considerably tested for their performance properties. This knowledge can be used when designing the architecture of a system and when choosing the components [27]. For components having to be newly developed or being hardly tested, the performance properties can be estimated with SPE methods. The resulting data can be taken into account for the component-based software engineering in the same way like that of a tested component.

Unfortunately, classical techniques for performance analysis are unsuited for the performance prediction of generic software components. They cannot cope with the parameterization and layering [23].

As a result, new performance analysis techniques have to be developed, specially made for the needs of component-based software engineering. An important aspect are the different contexts a component is to be deployed into. A static description of the performance of a component is impossible, because performance heavily depends on the context (platform, hardware, external calls, usage profile, etc.). Thus, the component has to be parameterized concerning its performance characteristics [20].

1.3 Contribution

The contribution of this work is twofold. Firstly, the applicability of the two performance prediction techniques CB-SPE and Palladio is empirically evaluated and compared from a developers point of view. Both techniques are not fully matured yet, and are not applied in practice. The work focuses (a) on the applicability of the techniques, (b) on the identification of potential

for improvement and (c) on the validation of assumptions made implicitly or explicitly by the methods.

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

- How correct are the performance predictions?
- What influence do the input models have?
- Is the workload for a performance prediction reasonable?

In section 3.2.2, the questions are further refined into metrics, using the Goal-Question-Metric approach [3]. By that, an empirical case study is designed. This design can also be applied to the empirical investigation of model-driven quality prediction approaches in general. This research method forms the second contribution. The empirical analysis to apply the metrics on has the form of a case study. A controlled experiment is desirable, but not accomplishable in the scope of this work.

2 Tested Performance Prediction Techniques

In this case study, two model-driven performance prediction techniques for component-based architectures are evaluated and compared. Both techniques are based on models of the later developed software, including data for the timing behavior of its components. To predict the performance, the software models are analyzed using methods of theoretical computer science (queuing networks) and mathematics (Fourier transformation), respectively, as opposed to other performance prediction techniques using simulations.

2.1 CB-SPE

The CB-SPE technique was developed by Antonia Bertolino and Raffaela Mirandola [4, 5]. It predicts and analyzes the performance of a system being assembled from components. CB-SPE is a compositional technique. It is based on the concepts of the well-known SPE and uses an RT-UML PA profile for input modeling.

The proceeding is compositional: At first, the component developer uses it at component level by making a parameterized performance evaluation of the single component. When composing the system, the systems assembler is provided with a step-wise procedure to predict the performance of the assembled components on the actual platform [5].

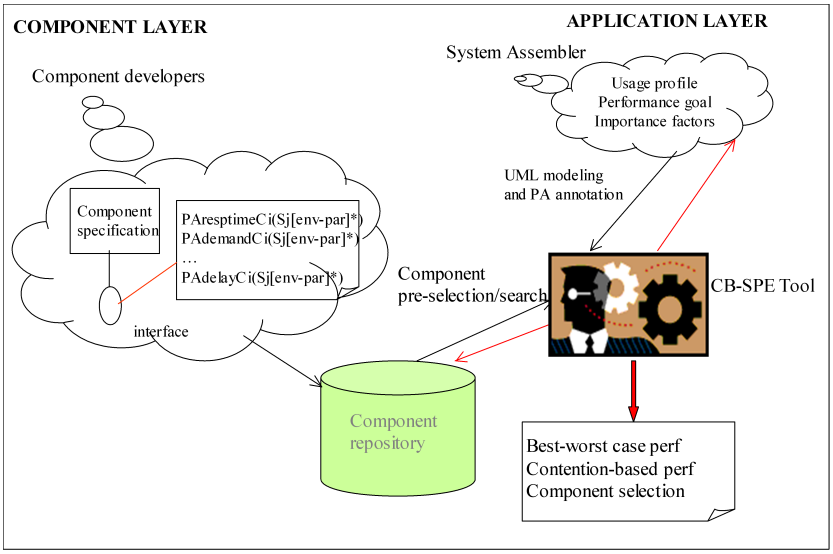


Figure 1: CB-SPE Framework (like presented in [5])

On the component level, the performance values for a component Ci's services S_j are analyzed for changing environment parameters $env-par$, The results are presented in a parametric form, $Perf_{Ci}(S_j[env-par]*)$, for the different performance indices *Perf* the component developer analyzed (examples for analyzed performance indices are demand of service, response time or communication delay). These parametric descriptions for the services of a component can be shipped with the component and independently used by the component assembler.

Step 1: Determine the Usage Profile
Step 2: Component Pre-selection
Step 3: Modeling and Annotation
Step 4: Analysis of Results

Figure 2: CB-SPE Procedure on Application Layer (System Assembler)

Figure 2 shows the proceeding on the application layer, i.e. the proceeding for the component assembler. First, the component assembler has to determine the usage profile. To do so, he has to analyze the different types of users and the required Use Cases.

Then, the component assembler preselects his components out of a set of offered components, each having parametric descriptions. To preselect, he can instantiate the parametric description of the services' performance indices he is interested in by inserting his particular environment parameters in the analysis results.

The components offering the best performance characteristics for his particular environment are further analyzed. Therefore, a description of the workflow of the system using sequence diagrams is created by the system assembler. The sequence diagrams objects represent the involved components. A deployment diagram is used to describe the available resources and their characteristics. Additionally, the number of users using the system concurrently can be

specified. In both diagrams, characteristics are specified using RT-UML PA annotations as well as extensions to RT-UML. Having completed the diagrams describing the workflow and the resources, the system assembler can use the CB-SPE tool to perform a performance prediction.

The CB-SPE tool gives a best-worse case analysis as well as a contention based result. With the best-worse case results, the component assembler can see whether his performance goals are feasible, i.e. lie between best case and worst case. If yes, the contention based results are considered. For the contention based results, the CB-SPE tool uses queuing networks, and thus include waiting times for contended results. In so doing, the behavior of the system with multiple users at the same time can be predicted.

With these results, the system assembler can decide whether the predicted performance values satisfy his needs or whether he has to look for other components that lead to better values.

2.2 Palladio

The Palladio technique [7, 8] is currently developed by the Palladio research group at the University of Oldenburg. This technique emphasizes (a) the parametrization of the specification of a component to be able to model their performance behavior for different contexts and (b) the use of distribution functions t describe the timing behavior.

According to [8], performance analysis not only needs compositional performance models to compute the performance of a system from the performance behavior of the single components. A component performance model is needed, too, modeling the performance of a single component. The performance values measured in one context, however, are not valid for other contexts, like mentioned above, and thus cannot be used for the prediction of the performance behavior in other contexts.

The technique uses a compositional component performance model. These should be compositional, parametric and as precise as possible. Therefore, it is based on parametric contracts. The response time (or other linear additive metrics, like reaction time) is specified by random variables in these contracts. Such a parametric component performance model is supposed to describe the dependency between the quality attributes of the component and the context [21]. Therefore, extended service effect automata are used. Service effect automata are finite state machines representing an abstraction of the control flow. The transitions of the automaton correspond with external calls, all internal computations are integrated in the nodes. The service effect automata are extended to Markov Models to describe the probabilities of execution of transitions and by adding a random variable to describe the timing behavior of internal and external services:

The resulting extended service effect automaton is a 7-tuple, $(S, \Sigma, T, s, F, P, B)$, consisting of

- a finite set of states (S)
- a finite set called the alphabet (Σ)

- a transition function ($T : S \times \Sigma \rightarrow S$)
- a start state ($s \in S$)
- a set of final states ($F \subset S$)
- a transition probability function ($P : S \times \Sigma \rightarrow [0, 1]$) and
- a function assigning timing behavior to transitions and states ($B : (S \times \Sigma) \cup S \rightarrow DF$ wit DF being the set of discrete distribution functions.)

No static variables are used to describe the timing behavior, as the timing behavior of internal as well as external services is never fix. The distribution of the random variables itself depends on the service effect automaton of the called service. To be able to use a computed distribution function as an input for another computation, like needed in this compositional approach, discrete distribution functions are used. Thus, the technique is not dependent on a specific class of statistic distributions. The distribution of the random variable is specified with the Quality of Service Modeling Language (QML) [10].

```
from service require Performance contract {
  responseTime {
    percentile 40 < 50 msec;
    percentile 60 < 100 msec;
    percentile 90 < 250  msec;
  };
```

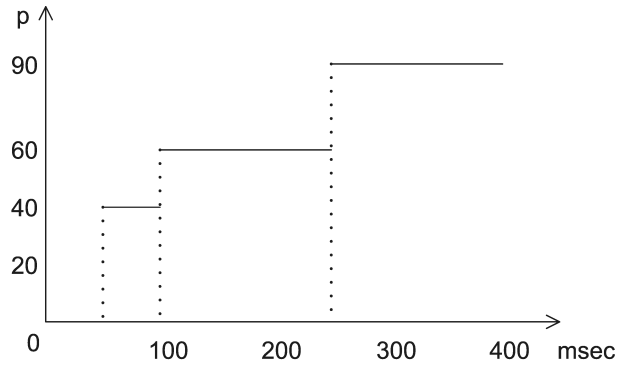


Figure 3: Palladio: Distribution Function Defined by a QML Contract

With the Palladio techniques, the distribution function of a service is computed based on the distribution functions describing the timing behavior of the internal computations and the external calls on the one hand and the control flow described with a Markov Chain on the other hand. To compute the resulting distribution function, the discrete Fourier transform is used.

By using this technique, statistical distributions of the response time of this component can be computed, taking into account the different execution possibilities. The resulting performance model is compositional: The computed distribution function for one service can be used as an input for the analysis of a second service calling this service.

The technique is not fully developed yet [8]. Questions of computational complexity and competing threads are an area of further research here.

3 Research Method

As mentioned in the introduction, the comparison of the two performance prediction techniques CB-SPE and Palladio is realized with a case study. Section 3.1 describes in details why a case study has been chosen and what is important about it.

Conducted without specific goals in mind, a case study can lead to a large amount of data. To extract the relevant information after collecting the data is hard, and it may be discovered that important information misses, because its relevance was not known beforehand. To be able to reduce this mass of data, eliminate irrelevant information and collect all relevant information, the goals of the case study should be worked out in advance. A well known and successful goal-oriented procedure is the Goal-Question-Metric Approach by Basili et al [3]. Section 3.2 describes the GQM approach shortly and introduces the GQM plan for this work, containing questions and metrics to compare the performance prediction techniques.

3.1 Case Study

When conducting an empirical analysis, the most convincing results are won by conducting a controlled experiment. With this form of empirical research, all factors that influence an experimental setup are controlled. Only the factors being the subject of the empirical analysis are varied (experimental variables or independent variable), all other factors are held constant. Thus, the changes of the results (the dependent variables) can be identified as caused by the changes of the experimental variables. 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 technique, whereas all other factors must be constant.

However, the effort for conducting a controlled experiment is high. The claim to controll all disturbance variables is hard to fulfill. Especially if humans are participating in the experiment, which is most often the case in the context of software engineering, there are many influencing factors. A large group of participants with a preferably equal knowledge is a good way to minimize or at least identify the influence of uncontrollable variables, because a strong influence of an individual's performance can likely be detected. If a strong effect is observed in the experiment, even a smaller group can lead to a high significance of the results. On the contrary, the smaller the expected effect, the bigger the group should be to get a high significance. The further investigation of these connections, however, is beyond the scope of this work. The participants have to be carefully chosen, to ensure that uncontrollable influences vary as little as possible.

For this empirical comparison of two performance prediction techniques, a controlled experiment is desirable. All factors except the used performance prediction techniques have to be controlled. Of course, a comparison of performance prediction techniques, and especially their applicability, involves human participants. It has to be tested whether the needed inputs for the techniques can be derived from the given information and whether the participants can handle

the techniques. To minimize the influence of uncontrollable variables, however, a large group of participants is needed. Within the context of this work, students will be asked to participate, and only about 20 participants are expected. Thus, the group is too small to be able to sufficiently eliminate the influence of uncontrollable variables. Additionally, the participants are not experienced with the used techniques, so a beforehand introduction is needed. But even with a beforehand training, it cannot be assured that the techniques will be equally known to the participants [14].

As a result, this empirical comparison will be conducted as a case study. In the context of software engineering, Prechelt [18] defines a case study as follows (and relates it to a controlled experiment):

> A case study is the description and evaluation of an instrument or a technique with a concrete example conducted ad hoc under artificial or typical conditions. Case studies may also compare several instruments or techniques, however, unlike controlled experiments, it is not guaranteed that all further factors are constant.

In this work, the compared techniques are the two performance prediction techniques. A concrete example is the performance prediction for a request to a prototypical web server. As not all factors have to be ensured to be constant, the experimental setup is less costly, and can be conducted within the bounds of this work. The drawback of a case study, however, is that the missing control makes it less generalizable. Only a single example system is used, and maybe the results here are not transferable to other architectures.

To achieve the best possible generalizability, I control the influencing factors as much as possible and use guidelines and techniques for controlled experiments like presented in [18]. When analyzing the results of the experiment, I try to identify the uncontrolled factors and interpret the results with this knowledge to assess the generaliziability.

3.2 GQM Plan

Primary principle of the GQM approach is that measurement should be goal-oriented. The first advantage of this proceeding is that having the goal in mind it is easier to choose useful and relevant data. This is supported by GQM's top down approach: On the basis of the goals, questions are found and further lead to metrics. The second advantage of the GQM approach comes with the backward proceeding: In a bottom up approach, the collected data is interpreted based on the questions and finally based on the goals [6]. The goals, questions and metrics together form the GQM plan.

There are several prerequisites for a successful use of the GQM approach [6].

1. The goal must specify with great detail what is to be analyzed.

2. Metrics have to be derived in a top down approach 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 will be regarded in this GQM plan, and will be explicitly named where fulfilled. Prerequisite 5, however, relates to appliance of the GQM approach in practical surroundings, e.g. in software development. In research, the participants of a case study are almost never involved in the design of the GQM plan. Thus, this prerequisite is invalid in this work.

3.2.1 Goal of the Case Study

A GQM goal specifies purpose of measurement, the object to be measured, the issue to be measured, and the viewpoint from which the measure is taken [3]. By naming all these parts of the goal, prerequisite 1 is fulfilled. Here, the GQM goals is to

> empirically compare and evaluate the applicability of the two model-driven performance prediction techniques CB-SPE and Palladio from a developer's point of view.

Note that the term developer means a developer of a system, who is at the same time the user of the two performance prediction techniques. Of course, it does not mean the developer of the performance prediction tool itself. The parts of the GQM goal are named explicitly in table 1.

Goal	
Purpose	Empirically compare and evaluate
Issue	the applicability
Object	of the two model-driven performance prediction techniques CB-SPE and Palladio
Viewpoint	from a developer's point of view

Table 1: Research Goal

3.2.2 Questions and Derived Metrics

In the following, the questions and metrics are derived based on the GQM goal, thus fulfilling prerequisite 2. A detailed rationale describing the metrics is given for each question, thus fulfilling prerequisite 3. Table 3 gives an overview on all derived questions and metrics. Questions and metrics are stated as generally as possible, to enable future experimenters to reuse them with other experimental setups.

For the following discussion, we introduce the following variables:

- Number of participants: m
- Number of participants in CB-SPE group: m_{CBSPE}
- Number of participants in Palladio group: $m_{Palladio}$
- Number of actions to assess a timing value for: n

1. How correct are the performance predictions?

   Rationale: The most obvious metric to answer this question is a comparison of the predicted performance and the actual performance of the implementation (metric 1a). For both techniques, the predicted average response time $predAvgRespTime_p$ is compared to the measured average response time $measAvgRespTime$ for each participant $p \in m_t, t \in \{CBSPE, Palladio\}$.

   To evaluate the deviation to the measured average response time, the absolute deviation $absDev_p = |predAvgRespTime_p - measAvgRespTime|$ is determined. Finally, the average absolute deviation over all participants applying one technique is assessed: $avgAbsDev_t = \sum_{p=1}^{m_t} absDev_p$ for $t \in \{CBSPE, Palladio\}$, For the Palladio technique, the predicted distribution function is compared to the distribution of the measured time consumption, too. To do so, the absolute deviation of the distribution functions is compared.

   However, the Palladio performance prediction technique does not claim to deliver absolute performance values, but to help comparing different design decisions at architectural level. Thus, to be right does not mean to predict the right response time or other performance metric, but to identify the design decision leading to the higher performance. If only two design options are available, the resulting metric is quite simple, as the percentage of correct design decisions $perc_t$ (metric 1b) for each technique t can be looked at:

   $$perc_t = \frac{\text{Number of correct decisions}}{m_t}$$

   However, for the more general case of more design options, the metric becomes more complicated. In [15], a solution is presented: Not only the identification of the best alternative, but the ranking of all design decisions should be taken into account. However, if two design options lead to almost the same response time, their order should have no

impact on the judgment of the ranking. Thus, the design options leading to similar results should be clustered, and the ranking of these clusters can be assessed.

What metric is actually more important in a performance engineering process depends on the requirements. If two alternatives are to be compared, the actual values may be less important than the result of which has the best performance. If one component has to be tested for its performance behavior, the actual values are wanted.

Metrics:

a) Comparison of predicted performance to measured performance.

b) Percentage of correct design decisions based on prediction: $perc_t$

Note that the rightness of the techniques is looked at in the context of this question. A further possibility is to evaluate the accuracy of the techniques, comprising both rightness and precision (c.f. section 6.3).

2. What influence do the input models have?

   Rationale: Assuming the used technique for computing the performance is correct, the results of a performance prediction depend on the given inputs. Therefore, it is interesting to analyze the influence of the inputs on the results and find the reason why a performance prediction is good or bad, i.e. whether it correctly predicts the performance or makes wrong predictions.

   It is likely that even a perfectly right performance prediction technique used within a prediction technique delivers wrong results if the used performance model reflects the actual system poorly. But an input model will not perfectly reflect a system in all its details, especially if it is based on estimations. Thus, the influence of impreciseness of the results has to be analyzed. This can be done by analyzing the techniques sensitivity, i.e. how much the results vary in relation to the statistical spread of the inputs.

   The first metric in this context describes the statistical spread of inputs, more precisely of the estimations. The participants will be required to write down their timing estimations for an action $i \in n$ during the experiment. With the term action, I subsume all blocks of computation and network delay that are relevant for performance prediction and that are assigned some estimated timing value when using the techniques. For CB-SPE, the timing estimation for an action $i \in n$ is the demand of computation or the network delay. For Palladio, the timing estimation for an action $i \in n$ is the time consumption of internal or external services.

   For each action $i \in n$, the average value avg_i of the estimated timing value over all participants is calculated (Note that the measured average response time is not of interest here, as this question only relate to the statistical spread of the input data, not to the rightness of the input data.) With this average, the absolute deviation from the average for each participant and each assessed action can be evaluated: $devEst_{i,p} = |est_{i,p} - avg_i|$ with $est_{i,p}$ being participant p's estimated timing value of action i. To get a single deviation value for each participant p, the previously calculated absolute deviation values $devEst_{i,p}$ are summed up: $devEst_p = \sum_{i=1}^{n} devEst_{i,p}$. As all participants estimated the time behavior for the same number of actions, $devEst_p$ does not have to be normalized. Looking at the distribution of these deviations $devEst_p$, the statistical spread of the estimations can be assessed (metric 2a).

Next, the statistical spread of the resulting performance predictions is looked at. For CB-SPE, the result of a prediction is the average time in the network, which can be used directly in this metric. The Palladio output, however, is a distribution function. To be able to compare the techniques outputs, the average value of this distribution function is used for this metric, too.

Metric 2b describes the statistical spread of the results. The metric is similar to metric 2a: First, I average over the predicted performance. With the average value $avgPred$ the absolute deviation of each participant's prediction from the average can be assessed: $devPred_p = |pred_p - avgPred|$ with $pred_p$ being the predicted performance of participant p. As with metric 2a, with the distribution of these deviations $devPred_p$, the statistical spread of the estimations can be assessed (metric 2b).

With metrics 2a and 2b, the sensitivity of the performance prediction techniques can be assessed. The sensitivity of a technique describes how susceptible it is to changes in the input. For the applicability of a performance prediction technique, it is best if the results are insusceptible to inaccuracy in the input data. Thus, the sensitivity must not be too high, as otherwise the results vary too much with similar inputs. To analyze the sensitivity of the techniques, I create a matrix containing the different combinations of metric 2a and 2b. The techniques will be classified in this matrix according to the results of the two previous metrics.

	Statistical spread of outputs: high	Statistical spread of outputs: low
Statistical spread of inputs: high	No statement about sensitivity is possible.	Sensitivity is low → good applicability
Statistical spread of inputs: low	Sensitivity is high → poor applicability	Sensitivity is alright, as is applicability

Table 2: Metric 2c: Classification Matrix to Assess Sensitivity

Note that for a high statistical spread of the input data and a high statistical spread of the results, it is impossible to assess the sensitivity, as it is unknown whether a low statistical spread of the inputs would result in a low statistical spread of the results, too. A low statistical spread of the inputs resulting in a low statistical spread of the results gives us more information. It is, however, not an optimal classification for a performance prediction technique, as it is unknown how the technique reacts with a high statistical spread of the inputs. The two remaining classes in the matrix deliver clear results: A technique that produces a low statistical spread of the results from a high statistical spread of the input data will likely also produce a low spread of outputs from a low statistical spread of the inputs. On the other hand, a technique that produces a high statistical spread of the results even from a low statistical spread of the inputs will likely also produce a high spread when confronted with wide spread inputs.

A further metric is supposed to analyze the difficulty of creating an input model. A performance predictions technique should include a way to easily and straightforward

create an input model or use existing ones. Here, the difference of the input models created by different participants can give information about how obvious input models can be extracted from given information. However, it is hard to state this difference in numbers, thus a scale from low to high will be used (metric 2d).

Metrics:

a) Statistical spread of inputs (i.e. of estimations).

b) Statistical spread of the output data (i.e. of predicted performance value).

c) Classification of the techniques sensitivity based on metric 2a and metric 2b.

d) Differences of input models on a scale low to high.

The evaluation of the influence of the input models can go much further. For a more elaborate evaluation and comparison, however, a more complex experimental setup is needed. To analyze the influence of single properties of the input models, for example, lots of data point only differing in a single property are needed. Additionally, the system to be analyzed in the experiment must be of sufficient complexity, using the techniques' possibilities as much as possible. Another further aspect that could be evaluated in the context of this question is the influence of the distribution functions the participants specify, e.g. the deviation caused by imprecise distribution functions.

3. Is the workload for a performance prediction reasonable?

   Rationale: To answer this question, two different groups of metrics are needed. First, the validity of the performance prediction results have to be taken into account. The effort for using a technique that does not deliver accurate predictions is likely to be unreasonable. On the other hand, if an intuitive prediction is much less accurate than a prediction with one of the techniques, the effort for using the technique is likely to be reasonable. Of course, the second group of metrics needs to be considered, too: The workload of a performance prediction technique must have reasonable workload toward an intuitive performance prediction. A very accurate performance prediction may not be useful if it comes with a high workload to predict the performance of a system.

   To evaluate this question, the workload for intuitive and technical performance prediction as well as the rightness of results have to be analyzed. The workload can be divided into acquainting oneself with the prediction technique on the one hand and performing the prediction on the other hand. The training time, of course, is less critical, as this workload only comes once. However, the training time is to be included. The time the participants needed to deal with the preparatory exercises can be viewed as a part of the training effort. It also helps to avoid subjective appraisal of the own training time: A participant A claiming to be acquainted with the technique after 5 minutes of training may need more time to solve the exercise than a participant B who familiarized himself for half an hour with the technique, as A was less prepared when starting with the exercise than B.

   To assess the time needed to train the techniques, not only the time measured for preparation, but the effectiveness of the training has to be considered. If a training is short, but the participants are not familiarized with the techniques afterward, it may have been too short. To measure the success of the beforehand training, the number of correct preparatory exercises as well as number of requested corrections for the experimental exercise

is considered. A participant who solves the preparatory exercise is likely to be well acquainted with the techniques. Additionally, the number of requested corrections for the experimental exercise shows how familiar the participants are with the techniques.

Metrics:

a) Metric 1a), for both techniques and intuitive proceeding.

b) Metric 1b), for both techniques and intuitive proceeding.

c) Time needed for the performance prediction in the experiment.

d) Time to become acquainted with the techniques.

e) Time to solve the preparatory exercises.

f) Number of correct solutions for preparatory exercises handed in.

g) Number of requested corrections for the experimental exercise.

Question 1	How correct are the performance predictions?
Metric 1a	Comparison of predicted performance to measured performance.
Metric 1b	Percentage of correct design decisions based on prediction.
Question 2	What influence do the input models have?
Metric 2a	Statistical spread of inputs (i.e. of estimations).
Metric 2b	Statistical spread of the output data (i.e. of predicted performance value).
Metric 2c	Classification of the techniques sensitivity based on metric 2a and metric 2b.
Metric 2d	Differences of input models on a scale low to high.
Question 3	Is the workload for a performance prediction reasonable?
Metric 3a	= 1a), for both techniques and intuitive proceeding.
Metric 3b	= 1b), for both techniques and intuitive proceeding.
Metric 3c	Time needed for the performance prediction in the experiment.
Metric 3d	Time to become acquainted with the techniques.
Metric 3e	Time to solve the preparatory exercises.
Metric 3f	Number of correct solutions for preparatory exercises handed in.
Metric 3g	Number of requested corrections for the experimental exercise.

Table 3: Summary GQM Questions and Metrics

There are other possible questions to compare the applicability of the techniques. The applicability of the used tools is essential for performance prediction techniques in practice. However, both evaluated techniques come with tools that are developed for research rather than practical applicability, and therefore the tools are prototypical. Object of this comparison are the underlying techniques themselves, not their existing implementation. Of course, an evaluation of the applicability of the tools should be carried out as soon as the tools are claimed to be mature.

At this point of time, an evaluation is reasonable, too. Early evaluation of the techniques is needed to (a) correct possible errors in the methods, (b) find points for further research and (c) detect wrong assumptions that have been made. By detecting these errors or missing aspects early, the techniques can be changed without having put much effort in the wrong direction.

4 Design and Conduction of the Case Study

This section describes the experimental setup, as well as the required preparations. As mentioned in section 3.1, the empirical study is conducted in form of a case study, because not all variables can be controlled. However, as effort is taken to control as many variables as possible, I try to achieve internal and external validity (c.f. section 6.1).

During the experiment session, two groups of participants are asked to analyze a prototypical web server for its performance, each applying one of the two performance prediction techniques. Hence, the performance prediction technique to be used is the independent variable. A third comparison group analyzes the system without any technique. To prepare the test group participants for the experiment, training sessions are arranged.

4.1 Participants

When designing an experiment, the participants are the first issue to consider. For this experiment, the students of the course ”Component-Based Software Engineering” at the University of Oldenburg in summer term 2005 are asked to take part. Thus, the participants of the experiment are students of 3rd and 4th year. A common objection to experiments in software engineering involving student participants is that the results cannot be transfered to ”real” software development. According to [18], 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 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 our study, the experience within a specific application domain is irrelevant. Besides, students have two benefit, opposed to software developers: (a) ”students know what they are capable of, while many professional developers do not, as they lack steady evaluation and, more importantly, comparison with their fellows” [W. Tichy, personal communication to R. Reussner, Software Engineering 2005 conference] and (b) students have a similar individual background. Hence, outliers due to individual performance are less likely.

To further assess the participant’s abilities for this experiment, a questionnaire was issued. The results of the questionnaire, characterizing the participants, can be seen in figure 4.

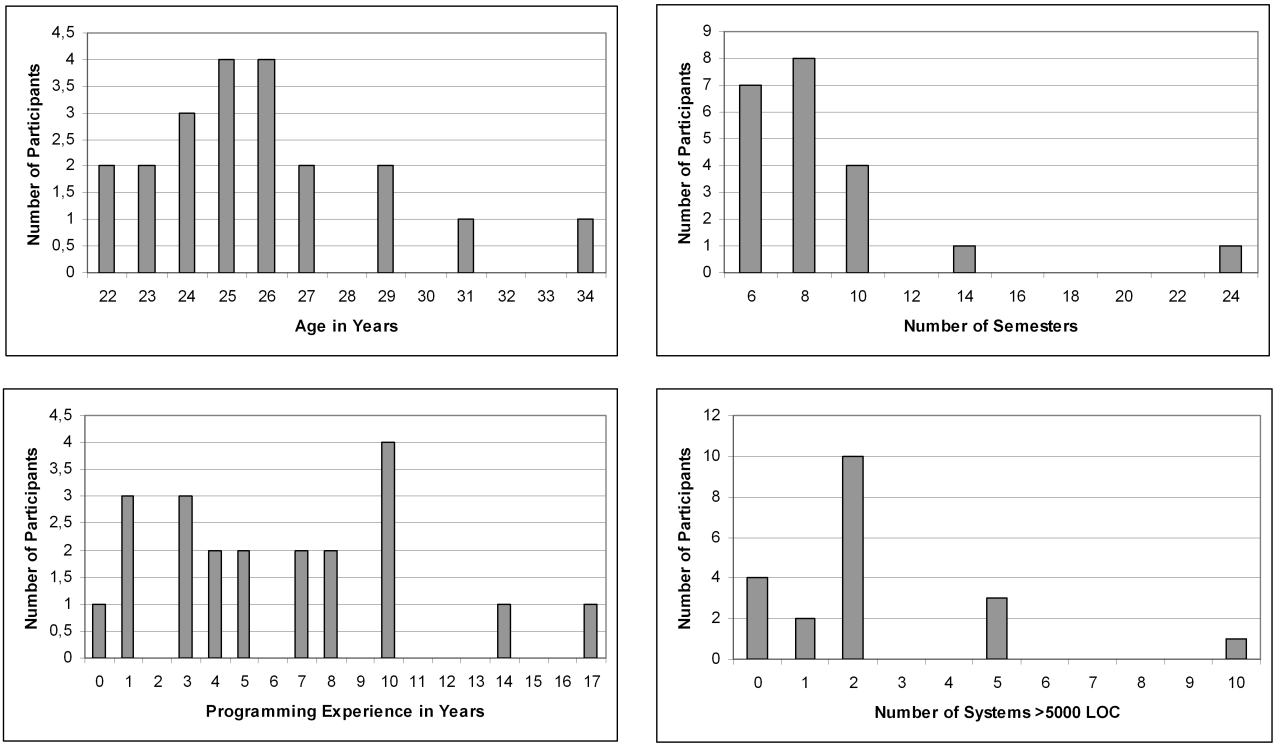


Figure 4: Information on the Participants

All participants have completed at least 6 semesters of study, thus they are advanced students. Additionally, they have participated at a software laboratory, as this is obligatory in the second year. Most participants have several years of programming experience, and designed one or more systems with more than 5000 lines of code.

Furthermore, all except two participants took the software engineering course, thus being familiar with UML notations. Both participants not having taken the software engineering course stated to have much programming experience. If UML was nonetheless not known to them, they did at least learn about it in the Component-Based Software Engineering course, which all participants have taken. Most participants had no or little experience with performance analysis. One participant stated to have medium experience with performance analysis, one stated to have much experience with performance analysis, but profiling and tuning only.

All things considered, the participants all have a base competence.Their competence can be compared to a competence of a professional whose main interest is not in performance analysis. Thus, the results of this experiment are transferable to the same situation involving professional software engineers, if not professional performance analysts.

The motivation or missing motivation of the participants can be a further problem for an experiment. To ensure that the participants take their tasks seriously, the participation is awarded. For each preparatory exercise, 15 points can be achieved. Not the correctness of the results, but the right applying of the techniques is awarded. Together, the achieved points make up 30% of the Component-Based Software Engineering course grade. The participance in the whole

experiment is nonetheless voluntary: Alternatively, the student may write a housework to get full marks in the course.

For the experiment, the award is different. The participants should not be under too much pressure, thus the possibility to achieve full marks in the course should not be connected to their success in the experiment session. On the other hand, to nonetheless achieve a certain motivation, up to 6 bonus points are awarded. Bonus points are added to the participants marks (on a scale 0 to 100) after having summed up the points for the preparatory exercises (0 - 30) and the oral examination (0 - 70).

For the comparison group, graduated computer scientists are asked to participate. Thus, the comparison group is at least as qualified as the two test groups. This may work against a possible effect of better predictions and design decisions by the participants. If such an effect is observed nonetheless, this observation can be quite relied on. However, the work of the comparison group cannot be as well controlled as that of the two test groups. Furthermore, only a smaller number of participants is available.

4.2 Preparation

The participants have to train the technique beforehand, as untrained participants would have to use a main part of the time in the experiment session to learn the techniques, thus leaving no time to actually work on the task. Additionally, problems with understanding the techniques can be discussed out beforehand. To ensure that the participants are familiar with the techniques, training session are established. For each technique, a tutorial session is held, presenting the technique and describing its appliance. Subsequently, a preparatory exercise is handed out, each participant has one week to work on. While working on the preparatory exercises, the participants can ask me for help. In a third tutorial session, a sample solution for each preparatory exercise is presented and the participants have again the opportunity to ask questions.

The preparation does not only train the participants, but tests their abilities as well as the formulation of the task and thus fulfilling the role of a pretest [18]. With a pretest, the learn effects during the experiment is minimized, as the participants learn during the pretest. Learn effects during the experiment itself may invalidate the results of the experiment. Additionally, the pretest can be used to assess the participant's abilities and balance the two groups (each applying one technique) so that the ability of the groups are about the same. In this experiment, the two experiment groups are set up based on the participant's results in the pretest, so that each group has stronger and weaker participants.

4.2.1 Preparatory Execises

The preparatory exercises can be found in the appendix. With both techniques, an imaginary system to find flights for a specific date and time is analyzed. The flight finder component first requests a list of available airline web services from a service broker. Afterward, it requests

all airline web services for available flights at the specified time. Finally, the available flights are sorted by price and for the three cheapest flights, further details are requested. The exercise comprises communication over network like the later experimental task. Additionally, it uses different control flow elements, like loops and alternatives.

For the CB-SPE technique, a sequence diagram describing the application work flow as well as a deployment diagram describing the deployment of the components on different servers has to be created by the participants with ArgoUML. The response times for the calls to the service broker and the web services as well as the times for the internal computations of the flight finder are given. The diagrams have to be annotated with RT-UML PA tags describing performance values for the different steps of the sequence diagram and the nodes of the deployment diagram.

For the Palladio technique, a service effect automaton describing the flight finder has to be created. Here, the response times for the calls to the service broker and the web services are given, but the times for the internal computations of the flight finder have to be estimated. The participants have to create an input file for the Palladio tool that contains the service effect automaton as well as the timing values for internal computations (estimated) and external calls (given).

With the second preparatory exercise a questionnaire is issued. The questions regard the participants' opinion of the techniques, and collects critique.

4.2.2 Results of the Preparation

23 participants handed in results for the CB-SPE technique, for the Palladio technique 22 did so. For the CB-SPE, many participants experienced problems with the interplay of ArgoUML and the CB-SPE tool. Only 12 participants were able to analyze the system. Although no estimations were involved, the predictions of the participants varied. The reason for this variation is not always identifiable, some diagrams seem to be identical, but lead to different results. Some participants with analyzable diagrams had small errors in their diagrams, e.g. they forgot a return transition, or did not model the control flow for alternatives and loops correctly. All these types of errors are due to a more complex exercise. The experimental exercise has a simpler control flow. The participants who could not analyze their diagrams mainly had syntactical errors. As a result of the problems with the preparatory exercise, the experimental task is adjusted: The ArgoUML diagrams are given, the participants do not have to create them themselves. Thus, syntactical errors in the created diagrams are prevented. Actually, this is quite realistic: In a design process, sequence and deployment diagrams are probably available. The problems of creating an ArgoUML diagram should not be subject to this experiment, and therefore should be left out.

The second exercise trained the Palladio technique. Here, 18 participants were able to analyze the system, four participants were unable to do so. One participant of those three had syntactical errors, one left his input file incomplete, for the other two participants the reason could not be found. 5 participants correctly modeled the system, although two of them had to comment out parts of their service effect automaton to be able to do the analysis. The other participants did not

model the service effect automaton correctly. Here, as with the CB-SPE technique, the loops and alternatives caused of errors. Although the sequence and deployment diagrams for the CB-SPE group are given, the Palladio participants have to create their service effect automata themselves. Sequence and deployment diagrams are usually present in a design process, however, service effect automata certainly are not. Thus, it is realistic to give the ready sequence and deployments diagrams, but not the service effect automata.

In the third tutorial session, the problems that occured were discussed. Most problems were due to the complex control flow of the preparatory exercise. As the control flow of the system analysis during the experiment is less complex, no similar problems are expected during the experiment. Many other problem were caused by syntactical errors. This source of error was discussed in the third tutorial session, too. Additionally, during the experiment, the participants will be allowed to ask for help when unable to analyze the system.

For the experiment session itself, participants were grouped; each group applying one technique. This grouping bases on the participants' results in the preparatory exercise. Thereby, two factors are decisive: First, the groups should be equally well-trained and successful in applying the techniques. Second, the participants should apply the technique they were more successful in during the experiment. Table 4 shows the points the CB-SPE group members achieved in the CB-SPE practice and the the points the Palladio group members achieved in the Palladio practice.

CB-SPE group members	15	15	15	15	14	14	14	13	13	10	10	9
Palladio group members	15	15	15	15	14	14	14	14	13	12	10	

Table 4: Achieved Points in Preparatory Exercises (of Respective Technique)

4.3 Experiment

4.3.1 Experimental Task

During the experimental session, the response time of a component based, prototypic web server should be analyzed. The web server was developed in the Palladio research group for testing purposes.

The server receives requests following the HTTP protocol and delivers the requested sites or documents. For each request, a thread is spawned, handling the request. First, the request is parsed by the `RequestParser` and its subcomponents, then forwarded to a corresponding request processor. For a HTTP request, the requests is parsed by the `HTTPRequestParser` subcomponent and forwarded to the `HTTPRequestProcessor` component. Here, the requested content is retrieved, either by retrieving a static file from the file system or by building the content dynamically. The retrieved content is then converted to a byte array. A component providing auxiliary functions (the `HTTPRequestProcessorTools` component) is used to send header and content to the client. All requests are saved in a log file after sending the file.

For this experiment, the response time of the server when requesting a static HTML document (size 50 kB) is analyzed. The involved components can be seen in figure 5. Note that the `WebserverMonitor` has been left out, as it is called after the content has been sent to the client, thus beyond the response time of the web server.

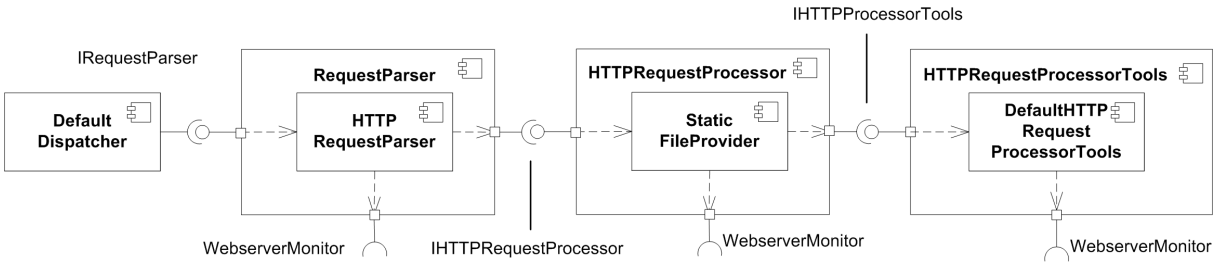


Figure 5: Involved Components of the Web Server

For comparing the performance prediction techniques, at least two different design alternatives are needed. Here, the original web server design is compared to a design in which the content is compressed before sent to the client. For the compression algorithm, existing libraries can be used. The decompression can be done by all current browsers. By analyzing the performance of both design alternatives, the participants have to find out whether more time is saved by sending smaller files then time is used for compressing the content.

For the design option with compression, a second subcomponent, `ZipHTTPRequestProcessorTools`, is inserted into the `HTTPRequestProcessorTools` component. The two subcomponents are now arranged in a Pipe-and-Filter pattern: Requests to the IHTTPRequestprocessorTools interface are first delegated to the `ZipHTTPRequestProcessorTools` component. The content is compressed using SharpZipLib, an open source compression library for .NET [13]. After compressing the content, it is forwarded to the `DefaultHTTPRequestProcessorTools`, who sends both content and header.

During the experiment session, the participants are asked to evaluate both design alternatives and identify the design option more advantageous for performance. This does intentionally not just ask for the lower response time, so that the participants may concern other values like utilization of the resources, too. Two groups are established: One group applies the CB-SPE technique, one applies the Palladio technique. All variables of the groups, except for the used performance technique, are held as constant as possible. A comparison group of graduated computer scientists is asked to analyze the system without any technique, but here the participants individual performance cannot be held as constant as for the two test groups.

For both techniques, the participants have to estimate the time consumption or workload, respectively, for certain actions (internal computation and external calls). To narrow down the values estimates by the participants, borders for this values are given. To get realistic borders, the web server is analyzed with the ANTS profiler by Red Gate Software [19], thus time consumptions for the different methods can be obtained. Of course, this profiling slows down the web server. However, the ANTS profiler states to subtract the time needed for profiling.

The exercises for the experiment itself were pretested by members of the Palladio research group. Thereby, problems with understanding and structuring could be removed. Additionally, the guidelines for the specification of the distribution function of the Palladio method were developed. This problem has not been visible in the preparatory exercises, as then distributions have been given and not created by the participants. The problem is to specify a certain, fix time consumption, as the Palladio tool understands distribution functions only. Therefore, a participant who estimates x milliseconds for a certain action to take cannot insert these x ms directly into the tool (as "in 100% of all cases, the action takes less or equal than x ms"), as it is interpreted as a distribution between 1 ms and x ms. However, it is impossible to specify lower bounds for a distribution. Thus, the distribution is artificial restricted by inserting the equivalent to "In 1% of all cases, the action takes less or equal than $(x - 1)$ ms. In 100% of all cases, the action takes less or equal than x ms" (Only integers can be inserted for defining the quantiles). Thus, in 99% of all cases, the time consumption for the action is right. The last 1% percent, however, cannot be controlled.

Both groups are given UML diagrams describing the system, as well as a explanation of these. The UML diagrams are conform to the UML 2.0 standard like presented in [9]. Additionally, information on the timing behavior is given: The average size of the file to be transfered is 50 kB and the speed of the network connecting the client to the web server is 1 Mb/s. Additionally, information on the time needed for calculations is given. The participants have to estimate the time needed for internal computations as well as the time needed for calls to the compression library and the component logging the actions. For both, a range is given to avoid very wrong estimations.

For the CB-SPE group, the sequence and deployment diagrams are given as an ArgoUML project, ensuring that no syntactical errors of the diagrams itself hinder the experiment (as mentioned in section 4.2.2). The given diagrams have to be annotated by the participants based on their calculations and estimations. Finally, they have to interpret the results.

For the Palladio group, the sequence diagram and deployment information are only given on the exercise sheet. To simplify the participants work, however, they are given an automaton comprising some of the service effect automata that have to be created when analyzing the system (see exercise sheet in section B of the appendix). As the first three components that handle the request have very simple service effect automata, their service effect automata are comprised in one automaton. This service effect automaton does not change with the design options. As a goal of the Palladio technique is to automatically derive service effect automata from other available information (e.g. sequence diagrams or byte code), this help does not invalidate the results.

For the comparison group, the exercise contains the same information as for the two test groups, i.e. the sequence diagram for work flow of the two design options and a text describing the work flow, the deployment of the component and the approximate bounds for the executions of different steps. However, no information on how to predict the performance is given.

Except for tool-specific information, the task is identical for both groups. No group has more information regarding the system to be evaluated than the other one. All differences of the tasks can be traced back to differences in the techniques, and thus are a part of the experimental

variable. The final exercise sheets, (for both techniques as well as the intuitive prediction of the comparison group) can be seen in section B in the appendix.

With the experimental task, a second questionnaire is issued. The questions regard the participants former experience, the time they needed to work on the preparatory exercises and the time they need to work on the experimental exercise. The results are used to assess the participants' abilities in section 4.1 and to answer question 3.

4.3.2 Execution of the Experiment

The experiment was conducted on June, 28th 2005 at the OFFIS research institute. Due to the limited space and PCs, the groups did not work concurrently: The CB-SPE group appeared in the morning, the Palladio group appeared in the afternoon. The participants worked at 6 desktop PCs and 6 notebooks, all having installed the required tools. The experiment was announced to take two hours, although participants who did not complete the analysis could continue after this time. Due to former experiences with this kind of experiments [14], most of the results were checked before the participants left. Thus, apparent errors could be detected and removed on the spot. However, some participants left without having their results checked. Possible errors by these participants have not been detected until the sight of all results, then, they have been asked to correct the solutions (included in the correction statistics).

After the experiment, the resulting data was collected and recorded in a spread sheet tool. More gross errors were found, that invalidates the results. The participants were asked to correct them before receiving the bonus points for the experiment. Two different kinds of errors have been detected.

- Falsely specified distribution: Most errors have been made in defining the distribution functions. Some participants did not adhere the notes on the task sheet. Others specified a distribution, but one that does not confines the distribution function, thus leading to the same results like with no restriction.
- Miscalculation: Some participants had errors in their calculations, e.g. calculating a wrong transfer time from a given network speed and file size.

A week after the conduction of the experiment with the test groups, the comparison group was asked to analyze the performance of the web server. Here, the exercises were mailed to the participants, and the participants worked on them alone. Thus, the variables influencing the participants could not be controlled. Five participants sent back solutions of the analysis.

Errors were also encountered in the solutions of the intuitive group. Here, one participant miscalculated the transfer time for a file, too. Thus, the errors are not due to distraction through the tools.

4.4 Problems

4.4.1 CB-SPE Problems

Beyond dispute, the biggest problem of the CB-SPE technique is the creation of the ArgoUML diagrams. The transitions are hard to handle and there is no undo function. Additionally, it often happens that a diagram cannot be analyzed by the CB-SPE tool for unknown reasons. As sequence diagrams are needed, ArgoUML 0.12 has to be used, being the last version supporting sequence diagrams. In later version, the sequence diagrams have been left out due to problems [25].

The CB-SPE tool comes with a few problems, too. Error messages are not in English, thus hard to understand. Additionally, they are quite imprecise. However, as the tool more like a prototype, these problems are not problems of the technique itself, but only of its current implementation.

Currently, a new version of the CB-SPE tool is developed, that should be integrated in the Eclipse IDE. Activity diagrams will replace the sequence diagrams, thus the modeling problem might be removed.

4.4.2 Palladio Problems

Here, the specification of the distribution function is the main problem. As mentioned in section 4.3.1, no lower bounds for distributions can be specified. Of course, all time consuming steps have some lower time bound. The limitation of the distribution function introduced to the participants on the experimental task sheet helps specifying the distribution, but not completely. Especially with high estimated times, the hundredth of the distribution function that cannot be constrained heavily influences the average of the distribution function. Of course, some statistical values like the average are more subject to outliers than other, for example the median, but nonetheless such results can be misleading.

A further problem are the error messages, too, as they are sometimes are imprecise, too. However, as with the CB-SPE tool, both problems are rather problems of the implementation than problems of the techniques themselves. As the distribution functions are supposed to be specified in QML, lower bounds can easily be specified, and only the implementation has to be able to handle them.

4.5 Measurement of the Web Server

The actual response time of the implementation of the web server is measured after conducting the experiment. A notebook with a 1.8 GHz Intel Pentium 4 Mobile processor and 256 MB RAM is used for the measurements. To generate the HTTP requests and measure the response

time, the ”Web Performance Trainer” (WPT) by Web Performance Inc. [26] is used, as it has been successfully used in a similar study [14]. During the measurements, only the operating system (Windows XP Home Edition), the Web Performance Trainer and the tested web server were running.

Like stated in the experimental exercise, the WPT simulates a 1 Mb/s connection. Different files having the size of 50 kB are requested. For the web server implementation without compression, only one file was requested, as the kind of file does not influence the response time. For the web server implementation with compression, five files are tested, each leading to a different compression rate. A plain text file (containing recurring content) can be compressed by 96.9%. A second plain text file, containing a randomly generated text with no recurring passages, can be compressed by 72.3%. Two executable files lead to compression rates of 55.3% and 36.4%, respectively. Finally, transfering a JPG file leads to a compression rate of 1.3%.

The WPT is set to request the respective file for 10 minutes, in each case waiting for the delivery of the file before requesting it anew. Thus, if a big file is transfered, less requests are made within 10 minutes. Table 5 shows how much requests have been made for each compression rate and the web server without compression.

Compression rate	1.3%	36.4%	55.3%	72.3%	96.9%	No compression
Number of requests	426	472	498	529	578	427

Table 5: Number of Requests during Measurement of the Web Server

Unfortunately, it turned out that the simulated network speed of the WPT is not exact. For example, for the implementation without compression, the simulated network speed is higher than required: In the first measurement, the average speed was 133.025 kB/swhich is 1.064 Mb/s. With a network speed of 1Mb/s, 50 KB take 409.6 ms to be transferred. With a network speed of 1.064 Mb per second, 50 KB take only 384.89 ms to be transferred. This difference is not just marginal and has to be considered when comparing the measured and predicted values.

A repetition of the measurement for the implementation without compression lead to a simulated connection speed of 132.41 kB/s, which is not considerably lower. Unfortunately, the WPT does not allow an exact specification of the speed of the connection, so the simulation cannot be conducted with the exact connection speed. It is assumed that for this setup, the connection speed cannot be properly set.

For the measurement of the implementation with compression, the network speed does differ from the input requirements of 1 Mb/s (= 122.07 kB/s), too. For the first four measurements, the deviation from the aimed network speed is low. For the compression rate of 96.908 %, the simulated connection speed was 93.29 kB/s in the first measurement. The measurement has been repeated to get a simulated connection speed of 102.14 kB/s. Further measurements lead to no higher connection speed, so this simulation is used. The actual network speed (like given in the output of the Web Performance Trainer), that will be used in further sections, can be seen in table 6.

Compression Rate	1.3 %	36.4 %	55.3 %	72.3 %	96.9 %
Actual Network Speed in kB/s	129.22	118.89	123.80	124.92	102.14

Table 6: Actual Simulated Network Speed

Additionally, the time consumptions obtained with the ANTS Profiler seems to be too high. For the compression of a 15 kB file, a time consumption of 15 ms has been measured by the ANTS Profiler. However, the results of the WPT measurements show a much smaller difference (2.57 ms) between the response time of the web server without compression and the response time of the web server with a compression rate of 1.3%. This cannot be explained with the slightly smaller file transfered, as this can lead to a time saving of 5.06 ms (1.3% of the response time of the uncompressed) and a remaining difference of 10 ms at the most (assuming the time for compressing is constant and not depends on the file size). Thus, the ANTS results have to be wrong, i.e. to high. They were used to help with the estimations on the sheets, thus the information on the sheet is misleading.

5 Results

In this section, the results of the experiment are described. The experimental setup that lead to these results is described in section 4 (Experiment). The resulting data of the experiment can be found in the appendix. With this data, the conclusions drawn in this section can be reproduced. The data is analyzed with the Goal Question Metric Approach (GQM), the metrics and question are presented in section 3.2.2 GQM Questions and Metrics. It is essential for using GQM that the resulting data is interpreted on the basis of the beforehand stated metrics and questions.

5.1 Correctness of the Techniques

The GQM question was stated in section 3.2.2:

> How correct are the performance predictions?

Two metrics have been specified to answer this question: First, the predicted performance is compared to the measured performance of the web server (metric 1a) and second, the percentage of correct design decisions is determined (metric 1b). To interpret the first metric, the measurement results of the web server implementation are needed.

5.1.1 Measurement Results

The setup for the measurement of the web server is described in section 4.5 (Measurement of the Web Server). Values have been rounded off. First, the average response time of the system is considered. The web server implementation without compression has an average response time of 389 ms for a 50 kB file. The resulting response times of the web server using compression are shown in Table 7 for different compression rates. The files used here have a size of 50 kB, too.

Compression Rate	Response Time in ms
1.3 %	392
36.4 %	268
55.3 %	192
72.3 %	118
96.9 %	25

Table 7: Average Response Times of the Web Server with Compression

It is clearly visible that using compression greatly reduces response time if a high compression rate is used. But even with a lower compression rate of 36.4 percent, the response time is reduced by 31.1 percent. It is also visible that the time needed to compress a file is very low, as the difference of the response time of the web server without compression and the response time of the web server with compression, using a compression rate of 1.3%, is very small. Thus, the time needed for compression is probably not higher than 10 ms.

Figure 6 shows the measured response times, plotted by the corresponding compression rate. With this graph, response times for other compression rates can be assessed.

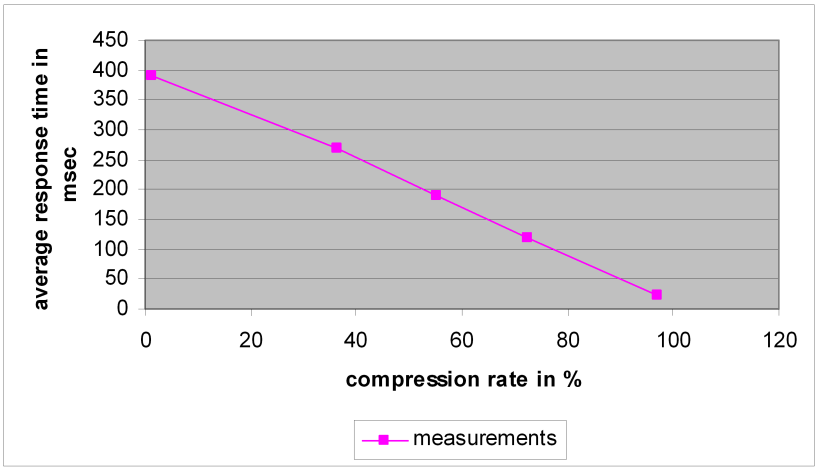


Figure 6: Average Response Time of Web Server with Compression

The distribution of the response times is shown in figure 7 and figure 8 (To associate the distributions with the compression rates in figure 8, note that the peaks have the same order as

the key below). The distributions are of different shape for the different compression rates. In general, they run out less steep to the right, toward higher values. The run of the curves is very uneven with peaks about every 10 milliseconds. As the common timer tick on Windows XP machines is 10 milliseconds [16], the peaks are likely to be results of these time slices. The actual distribution of the response time probably is more even, but cannot be measured.

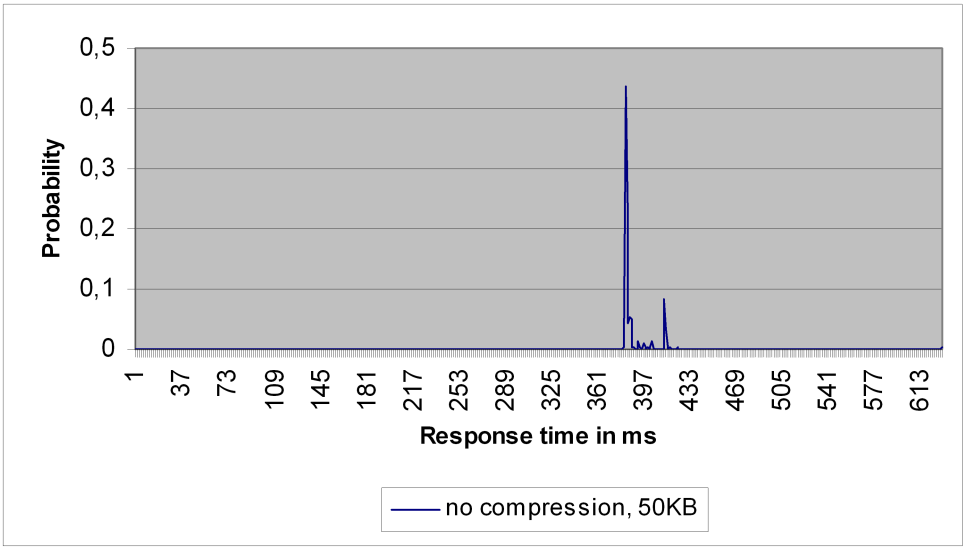


Figure 7: Distribution of Response Time of Web Server without Compression

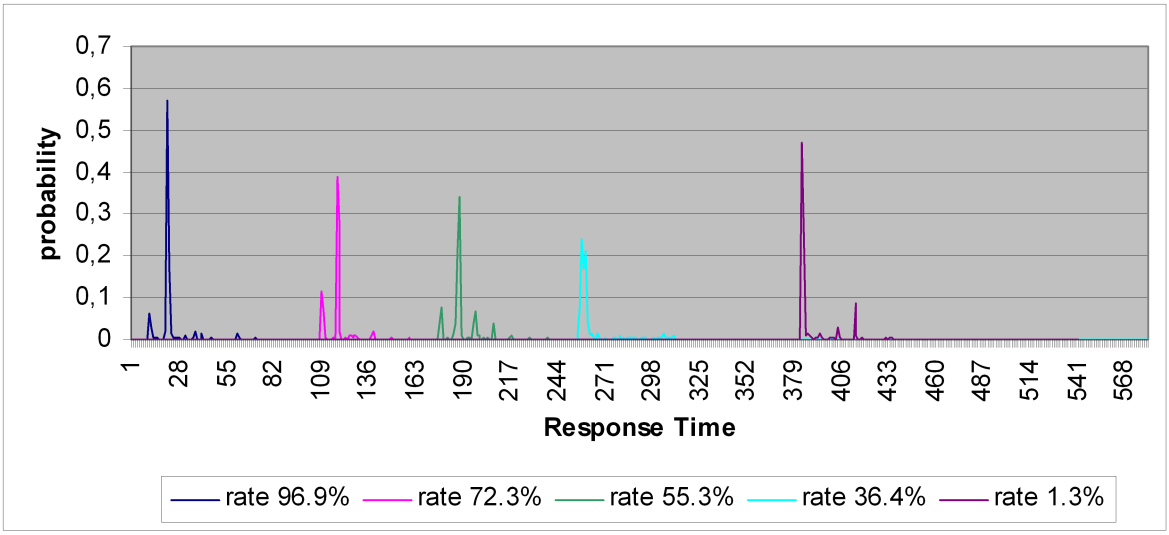


Figure 8: Distribution of Response Times of Web Server with Compression for Different Compression Rates

To be able to compare the measured distributions to predicted distributions, descriptive statistical measures are shown in table 8.

Compression rate	0%	1.3%	36.4%	55.3%	72.3%	96.9%
Average in ms	389	392	268	191	118	23
Variance in ms^2	44669.5	195951.8	297823.3	26719.6	10276.5	4653.6
Standard deviation in ms	211.35	442.66	545.73	163.46	101.37	68.22
Absolute deviation in ms	8.55	10.98	13.62	6.93	4.17	4.74
Lower Quartile in ms	383	384	258	187	118	21
Median in ms	384	385	260	188	118	21
Upper Quartile in ms	386	388	262	194	119	22
Difference lower quartile and median in ms	1	1	2	1	0	0
Difference upper quartile and median in ms	2	3	2	6	1	1

Table 8: Statistical Values of Measured Distribution of Response Times

Adjustment of Measured Data As mentioned in section 4.5, the network speed was not precisely set by the Web Performance Trainer. To eliminate the failure in the results caused by the improper connection speed , the results of the measurement are adjusted. To do so, I calculate the time needed for transferring a file of the corresponding size with both simulated and required connection speed. The difference is the time that has to be added to or subtracted from the average response time of the system. The resulting adjusted average response times can be seen in table 9, the column with compression rate of 0 % representing the measurement of the implementation without compression. The adjustment is the value that has to be added to the average response time. It is calculated as follows:

$$adjustment = \frac{fileSize}{reqNetSpeed} - \frac{fileSize}{simNetSpeed}$$

with $fileSize$ the size of the transferred file in KB, $simNetSpeed$ the simulated connection speed in kB/s and $reqNetSpeed$ the required connection speed of 122.07 kB/s. The resulting adjustment is in seconds, in the table it is given transferred to milliseconds. The adjustment may, of course, be negative if the simulated connection speed is higher than the required.

compression rate	0 %	1.3 %	36.4 %	55.3 %	72.3 %	96.9 %
measured av. response time in ms	389	392	268	191	118	23
simulated network speed in kB/s	133.02	129.22	118.89	123.80	124.92	102.14
file size in kB	50	48.83	30.37	22.46	13.87	1.55
adjustment in ms	33.73	22.15	-6.66	2.58	2.59	-2.47
adjusted average response time	423	414	261	193	121	20

Table 9: Adjusted Average Response Times of the Web Server with Compression

Figure 9 shows the adjusted response times, plotted by the corresponding compression rate.

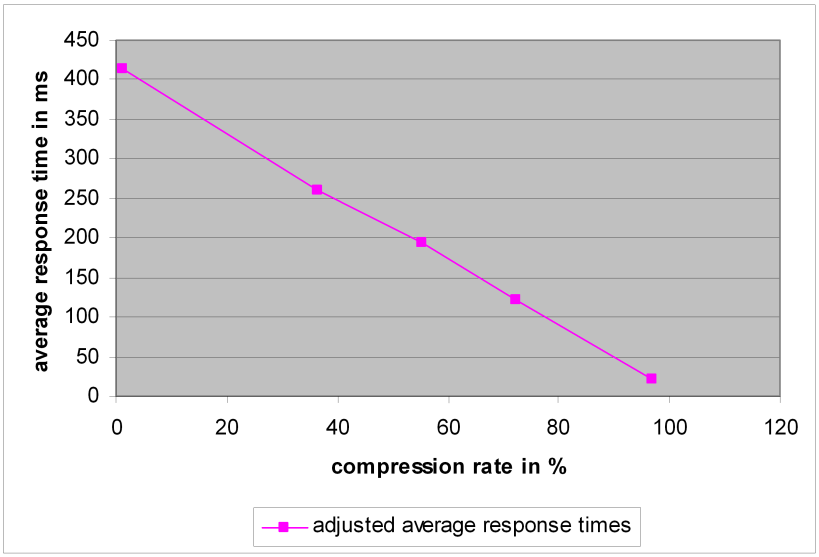


Figure 9: Adjusted Average Response Times of Web Server with Compression

In the following sections, the adjusted data will be used to compare and evaluate the performance predictions, except for the comparison of the distribution of the response time. For the comparison of the distributions, the original values are used, as the adjustment of the distributions is more delicate. However, for the comparison of the distribution functions, the mean response time is less important, as especially variance and deviation are looked at. Of course, it is problematical to change the measured values belatedly. When doing so, the predictions of the participants cannot be directly compared to measured values, but have to be compared to values that underwent another procedure. With showing that this procedure does not invalidate the results, however, the rightness of these adjustments is shown.

The adjustment of the values is quite simple. When delivering a file, the web server displays the exact file size of the sent file in bytes. Additionally, the Web Performance Trainer logs the actual bandwidth used during benchmarking the system. Thus, the time for transferring the file with the actual bandwidth as well as the time for transferring the file with the required bandwidth can be calculated. The difference of the two resulting values is added to the measured response time, to get the adjusted response time. Thus, the time measured for the calculations of the web server remains the same, only the transfer time is adjusted.

There is one risk in this calculation: It depends on the Web Performance Trainer detecting the right actual bandwidth. However, as mentioned above, the bandwidth was certainly not simulated correctly, as the response time is too low to send a 50 kB file with a 1 Mbit connection.

Derivation of Response Times for Other Compression Rates For the predictions of the web server with compression, the compression rates estimated by the participants are not equal to those measured. Thus, I derive the actual response times for these compression rates

from a spline fitting the measured values using first-degree polynomials. For each section of the graph, the function is determined. The resulting function is

$$respTime(x) = \begin{cases} -4.3471x + 419.3, & x \in (0, 1.3] \\ -3.5920x + 391.8, & x \in (1.3, 36.4] \\ -4.2472x + 428.1, & x \in (36.4, 55.3] \\ -4.0896x + 416.7, & x \in (55.3, 100) \end{cases}$$

The values are rounded off for clarity, in further calculations the precise values are used. The resulting response times for the compression rates used by the participants are shown in table 10. A variant would be to approximate a single linear function based on the measured values, but as the spline is almost linear, this will not improve the fitting.

Compression Rate in %	10	20	30	35	40	50	60	80
Derived Response Time in ms	375.85	332.38	288.9	267.17	248.14	212.22	173.23	89.5

Table 10: Derived Response Times For Compression

5.1.2 Comparison of Predicted and Measured Performance

CB-SPE 10 participants handed in results computed with the CB-SPE technique. Figure 10 shows their predictions for the implementation without compression as well as for the implementation with compression. All predictions for the web server without compression are close to each other, and range between 402 and 431 milliseconds. The absolute deviation here from the adjusted measured average response time ranges from 1.84 ms to 20.82 ms. The average absolute deviation from the adjusted measured response time is 7.97 ms.

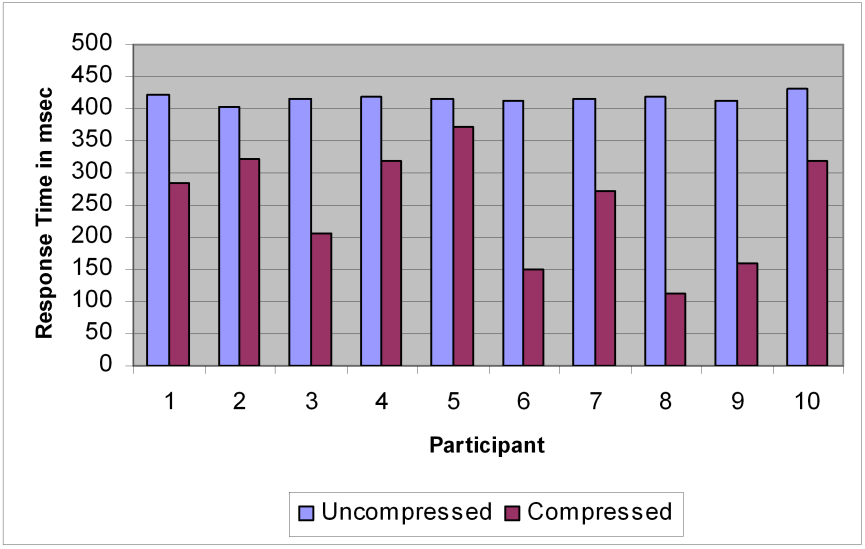


Figure 10: Predicted Response Times with CB-SPE Technique for Web Server

The predictions for the design option with compression are more spread, ranging from 112 to 373 milliseconds. This greater spread is caused by the participants estimating different compression rates. In section 5.1.1, the actual response time for the different compression rates estimated by the participants are determined from the measured response times. Both predicted response times and adjusted measured response time of the derived function can be viewed in figure 11.

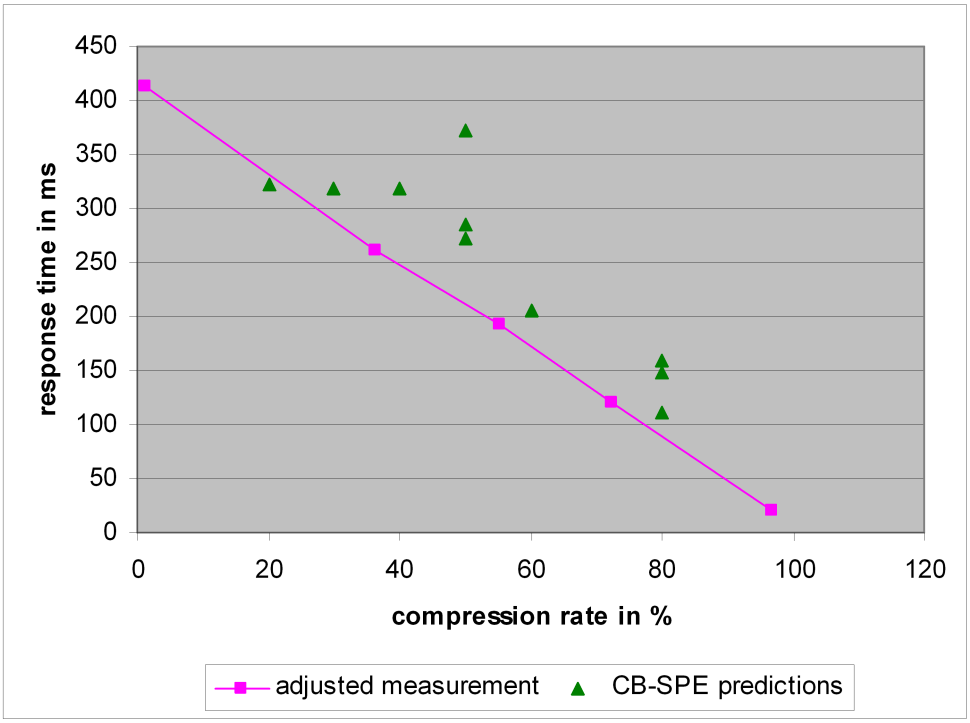


Figure 11: Comparison Adjusted Measured Response Times and CB-SPE Predicted Response Times for Web Server with Compression

With the derived function and the resulting response times for selected compression rates (table 10), the absolute deviation of the predicted response times can be calculated. The results range from 10.37 ms to 160.78 ms, with an average of 58.79 ms.

Palladio 11 participants handed in results computed with the Palladio technique. Figure 12 shows their predicted average response time for the implementation without compression as well as for the implementation with compression. The predictions for both design options are quite spread. The predicted average response time for the design option without compression ranges from 285.8 ms to 419.68 ms. The absolute deviation here from the adjusted measured average response time ranges from 3.16 ms to 137.04 ms. The average absolute deviation from the adjusted measured response time is 60.04 ms.

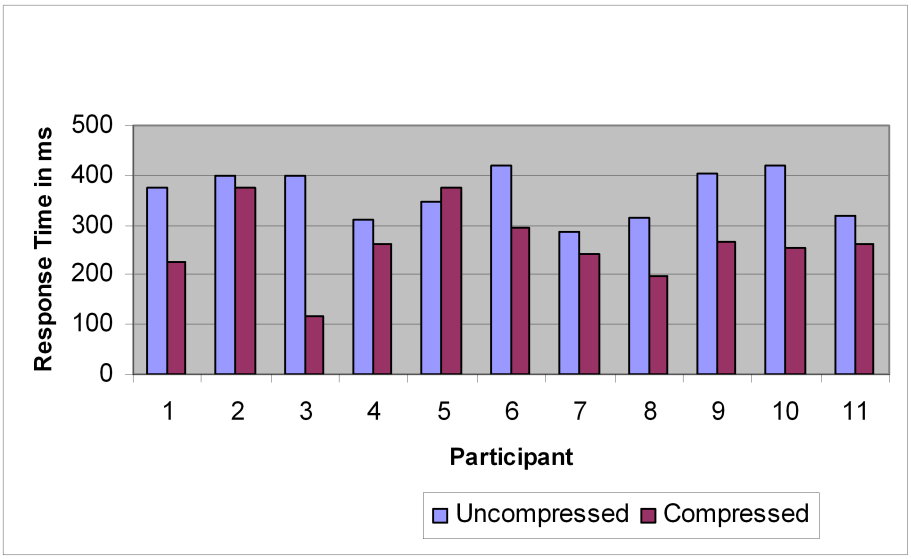


Figure 12: Predicted Response Times with Palladio Technique for Web Server

The predictions for the design option with compression are more spread, too, ranging from 118.75 ms to 376.35 ms. Like above, this greater spread is caused by the the participants estimating different compression rates. Both predicted response times and adjusted measured response time of the derived function can be viewed in figure 13.

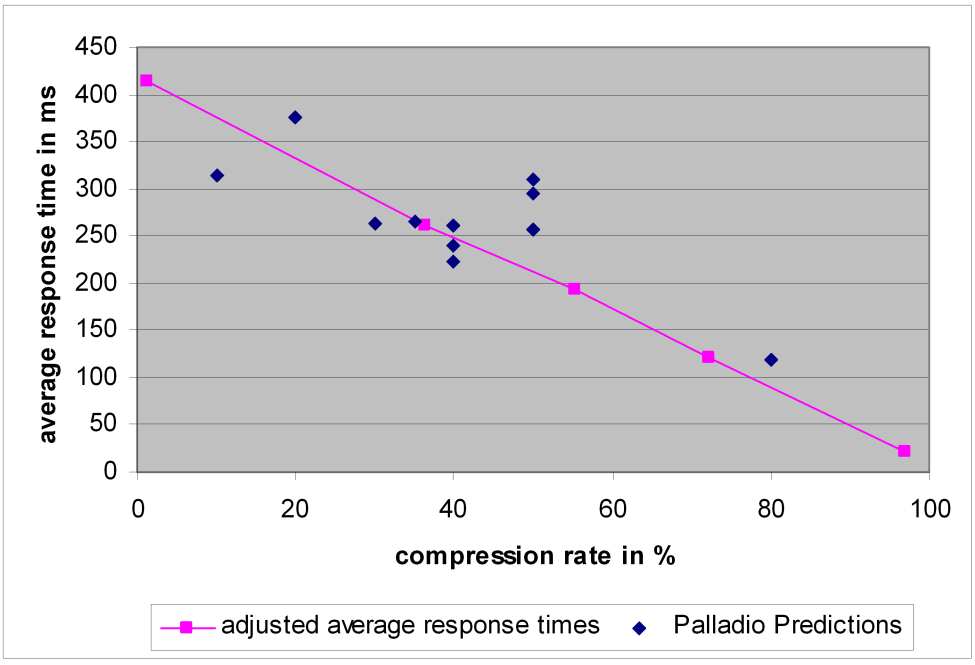


Figure 13: Comparison Adjusted Measured Response Times and Palladio Predicted Response Times for Web Server with Compression

Again, with the derived function and the resulting response times for selected compression rates

(table 10), the absolute deviation of the predicted response times can be calculated. The results range from 0.85 ms to 83.32 ms, with an average of 26.2 ms.

For Palladio, the predicted distribution is considered, too. The measured distributions of the response times of the web server for both design options can be seen in section 5.1.1 (figure 7 or 8, respectively). An excerpt of distribution for the design option without compression, predicted with the Palladio technique, is seen in figure 14. Figure 14 shows that the predicted distributions vary greatly, in table 11, the corresponding absolute deviations are listed.

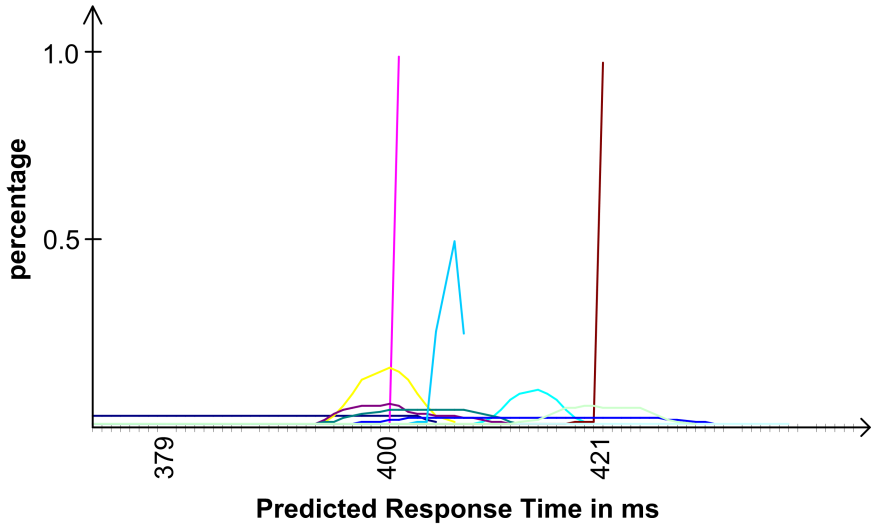


Figure 14: Excerpt of Distribution of Predicted Response Time for Web Server without Compression

The variations are the result of the different distributions the participants use within their service effect automata. As mentioned in section 4.3.1, the participants had to constrain the distribution of their estimated values, but were free to decide how to constrain the distribution. Participant 2 and 6, having the steepest distribution, constrained all distributions (external as well as internal computations) as tight as possible, i.e. they inserted every estimation of x ms as the distribution $((1, 100), (x - 1, x))$. Thus, in all cases, the execution of this step takes less or equal than x ms, and only in 1% of the cases, the execution of this step takes less or equal than $x - 1$ ms, so 99% of the cases have an execution time of exactly x ms. All other participants were less strict: Participant 9, having the third steepest distribution, allowed an interval of 2 ms for two of his estimations. Participant 3, having the forth steepest curve, thrice allowed an interval of 5 ms.

To compare the distribution functions, their absolute deviation as well as their quartiles are considered.

Table 11 shows the absolute deviations, the median and the quartiles of the different participants' predicted distribution functions, for both design options. Table 12 is an excerpt of table 8, showing the absolute deviation, the median and the quartiles of the measured distribution functions.

For the design option without compression, the average of the absolute deviations is 74.3 ms, whereas the absolute deviation of the measured distribution function is only 8.6 ms. For the

design option with compression, the average of the absolute deviations is 71 ms, whereas the absolute deviations of the measured distribution function range between 4.7 ms (compression rate 96.9%) and 11 ms (compression rate 1.3%).

Participant	Absolute deviation in ms	
	No compression	Compression
1	14.	11.1
2	4	1
3	4.5	44.9
4	114.4	96.3
5	110.9	116.2
6	4	2.8
7	130.6	107.4
8	115.6	114.6
9	4	76.4
10	200.4	116.8
11	114.9	93.7
Average	74.3	71

Table 11: Absolute Deviation of Predicted Distribution Functions

Compression rate	0%	1.3%	36.4%	55.3%	72.3%	96.9%
Absolute deviation in ms	8.6	11	13.6	6.9	4.2	4.7

Table 12: Absolute Deviation of Measured Distribution of Response Times

Measures for the dispersion are meaningful enough for normal variables. Here, the shape of the distribution is interesting, too. To compare the runs of the distribution functions, the upper and lower quartiles are considered. To assess the run of the distribution functions, I use the difference of the upper and lower quartile, respectively, from the median. The average of the differences between the quartiles and the median of the predicted distribution functions are shown in table 13. The differences between the quartiles and the median of the measured distribution functions can be seen in table 8, ranging between 0 ms and 2 ms for the difference of lower quartile and median and between 1 ms and 6 ms for the difference of upper quartile and median. It is detected that the runs of the measured distribution functions are less steep to the right, toward higher values, however, the runs of all predicted distribution functions are less steep to the left, toward lower values. Additionally, the intervals between the quartiles is much greater for the predicted distributions.

	No compression	Compression
Difference lower quartile and median in ms	140.2	119.8
Difference upper quartile and median in ms	31.8	27.5

Table 13: Difference Between Lower and Upper Quartile toward Median of Predicted Distribution Functions

5.1.3 Percentage of Correct Design Decisions

From the results of the measurements, it can be seen that compression reduces performance, however, the amount heavily depends on the compression rate.

The average compression rate the participants used is 46.9%. Of course, this depends on the contents the web server provides. The experimental exercise states that the server is supposed to host many packed images, that usually cannot be further compressed or even result in bigger files after compression. A photo gallery will likely have a worse average compression rate. To get practice values, I analyzed the starting site of Spiegel.de[1]. 85 files are transferred when opening the site. Here, the average compression rate (of course weighted by file size) is 59%. I also analyzed an image search with google, looking for "Hintergrund" (background)[2]. For the page showing the first 20 results, an average compression rate of 24.3% can be achieved. Viewing a small photo gallery of the student representatives of the Oldenburg Department of Computer Science [3] results in a compression rate of 9.3%.

It becomes clear that the statement ”The server hosts many packed images” does not imply a specific compression rate, but a wide range. The compression rate lies between 9.3% and 59% in the analyzed sample sites. The average compression rate the participants estimated is realistic, as it is within this range. However, many of the participants chose a compression rate of 80%, as this number is mentioned in the excercise. This is unrealistic: The exercise indeed states that with compression, a site can be compressed up to 80%, but at the same time it says that this server hosts many packed images.

The correct design decision for all compression rates the participants chose (starting from 10%) is the use of compression. All participants using the CB-SPE technique to predict the performance have correctly chosen this design option, thus the percentage of correct design decisions is 100%. All except one participant using the Palladio technique chose the design option with compression, thus the percentage of correct design decisions here is 90.9%. The one participant choosing the wrong design decision used the lowest compression rate of all participants, i.e. 10%. No participant using CB-SPE chose a compression rate of only 10%, so it is unknown whether it would have lead to a wrong decision, too.

Probably, this one participant chose the wrong design options because of the high time consumption for the compression of a 15 kB file given on the sheet (c.f. section 4.5) and the participant’s estimation for the time consumption for the compression of a 50 kB file, respectively.

Technique	Percentage
CB-SPE	100%
Palladio	90.9 %

Table 14: Percentage of Correct Design Decisions

[1] www.spiegel.de, accessed July 27th, 2005

[2] images.google.de/images?svnum=10&hl=de&lr=&q=Hintergrund&btnG=Suche, accessed July 27th, 2005

[3] http://www.fachschaft-informatik.de/uni-oldenburg/bildergalerie.php?verzeichnis=O-Woche%202004&galerie=Dienstag&oberdatei=ersti, accessed July 27th, 2005

5.1.4 Answers to Question 1: How correct are the performance predictions?

When comparing the predicted values to the measured values, it has to be taken into account that the time needed to compress a file was lower than provided on the sheet, c.f. section 5.1.1. Thus, a deviation toward higher values is likely to be result of this misinterpretation, and may be correct if the right guidelines had been set.

For CB-SPE, the predictions for the design alternative without compression deviate only by 7.97 ms, i.e. here the performance is reliably predicted. The results for the design alternative with compression are less good, most participants predicted a response time that is too high. However, this may be caused by the wrong information on the time to compress a file. Despite the impreciseness here, the correct design option has been chosen by all participants. The CB-SPE technique can therefore be called right.

For Palladio, the predictions for the design alternative without compression deviate more, the average absolute deviation is 60 ms. Here, it has to be taken into account that the Palladio technique delivers a distribution function in first place, and does not claim to deliver exact time amounts. The main goal is to be able to identify the better design decision. Additionally, the specifying of the distribution function in the inputs for the Palladio technique causes more spread, as the participants specified these differently. The inability of the tool to work with lower bounds for performance results in a distribution where a response time of 1 ms is always possible. Thus, the the mean of the distribution is further invalidated, as it is susceptible to outliers and no upper outliers are present to balance the influence of the outliers. The Palladio results for the design option with compression are better, even having a lower average deviation than the CB-SPE predictions (26.2). Here, the lower mean response time of a Palladio prediction balances the higher time estimated for compressing a file.

As the Palladio technique delivers distribution functions, too, these are compared to the measured distributions, too. For both design option, the absolute deviation of the majority of the predicted distribution functions is too high. For the design option without compression, the average of the absolute deviation of the predicted distribution functions is even ten times the measured one. Analyzing the quartiles supports this impression: The dispersion of the predicted distribution functions is too high. Additionally, the measured distributions run out less steep toward higher values, whereas the predicted distributions run out less steep toward lower values. This may be a result of the inability to specify lower bounds with the Palladio tool.

Even if the distribution of the measured response times could not be correctly predicted, there is a distribution in the measured response time, which is not reflected by the CB-SPE technique at all. I suppose that to get better distribution functions, more detailed instructions to the participants how to create a distribution are needed. This aspect has been unattended in the preparation. Additionally, the Palladio tool has to offer more powerful methods to define a distribution, as some of the participants' distribution functions are rather accidental, resulting from the inability to specify a lower bound for time consumptions.

The Palladio performance prediction is less correct than the CB-SPE predictions in this experiment. A reason may be that the tool is less well developed, being unable to specify lower

bounds. Additionally, the specification of distribution functions was problematically. The exercise did not contain information on the distribution of times. Thus, to further evaluate the rightness of the Palladio performance prediction technique, further experiments should take place, putting the stress on the specification of distribution functions and their influence on performance predictions.

5.2 Influence of Inputs

The GQM question for this area was stated in section 3.2.2:

> What influence do the input models have?

Four metrics have been specified to answer this question. The first three metrics, namely the statistical spread of the inputs (metric 2a), the statistical spread of the output (metric 2b) and the classification of the techniques sensitivity (metric 2c) are closely related and are analyzed in section 5.2.1. The forth metric, namely the differences of the input models on a scale low to high (metric 2d), is analyzed in section 5.2.2.

5.2.1 Classification of the Sensitivity of the Techniques

First, the metrics 2a and 2b are analyzed separately, each for both design options. Afterward, the actual classification of the techniques is carried out.

Metric 2a, design option without compression The summed deviation value $devEst_p$ for each participant is shown in table 15. $devEst_p$ is calculated like described in section 3.2.2:

$$devEst_p = \sum_{i=1}^{n} |est_{i,p} - avg_i|$$

(with $est_{i,p}$ being participant p's estimated timing value of action i, avg_i being the average estimated timing value of action i, n being the number of participants). Figure 15 and 16 show the distributions of the summed deviation values graphically. Statistical values for both distributions are shown in table 16.

CB-SPE											
Participant	1	2	3	4	5	6	7	8	9	10	
$devEst_p$	13.5	11.9	9.7	11.9	9.7	5.9	5.3	10.5	6.3	9.1	
Palladio											
Participant	1	2	3	4	5	6	7	8	9	10	11
$devEst_p$	5.7	19.1	13	8.1	18.6	11.5	13.9	10.2	13.7	8.4	6.4

Table 15: Summed Absolute Deviations of Each Participant's Estimations (Design Option Without Compression)

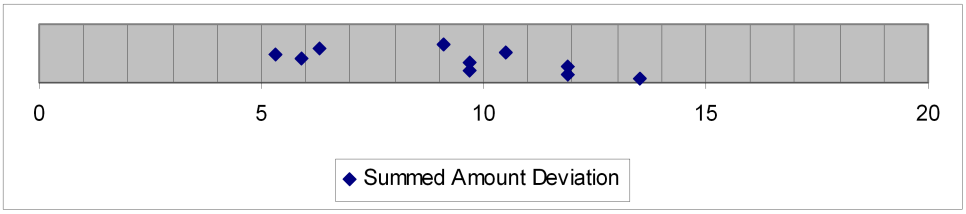


Figure 15: Distribution of Summed Absolute Deviations of Each Participant's Estimations (CB-SPE Technique, Design Option Without Compression)

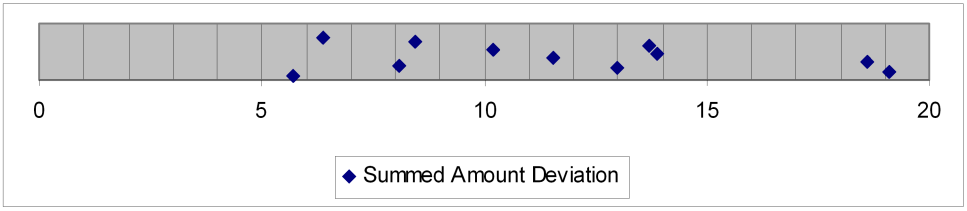


Figure 16: Distribution of Summed Absolute Deviations of Each Participant's Estimations (Palladio Technique, Design Option Without Compression)

Technique	Average	Average Absolute Deviation	Variance	Standard Deviation
CB-SPE	9.4	2.4	7.7	2.8
Palladio	11.7	3.6	20.3	4.5

Table 16: Statistical Values of $devEst_p$ Distribution (Design Option Without Compression)

For both techniques, the spread of the input data is present, but not too high. Note that the range in the figures 15 and 16 is quite small. The spread of the input data for the CB-SPE technique, having a lower mean and a lower deviation, is lower that that of the Palladio technique, but not significantly. Therefore, I classify the spread for both techniques as being relatively low.

Metric 2a, design option with compression For the estimations for the design option with compression, the metrics 2a and 2b are analyzed, too. Table 17 shows the summed deviation value $devEst_p$ for each participant. In figure 17 and 18, the distribution of these summed deviation values is shown graphically.

Statistical values for both distributions are shown in table 18. The spread of the inputs of the CB-SPE technique is lower, too.

CB-SPE										
Participant	1	2	3	4	5	6	7	8	9	10
$devEst_p$	137.8	60.8	40	38.6	40.8	37.6	19.6	68.4	43.6	46.8
Palladio										
Participant	1	2	3	4	5	6	7	8	9	10	11
$devEst_p$	8.9	45.4	60.3	21.3	62.3	26.4	25.1	31.5	29.4	28.7	20.1

Table 17: Summed Absolute Deviations of Each Participant's Estimations (Design Option With Compression)

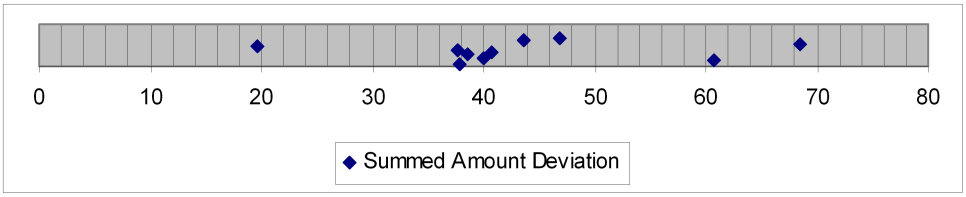


Figure 17: Distribution of Summed Absolute Deviations of Each Participant's Estimations (CB-SPE Technique, Design Option With Compression)

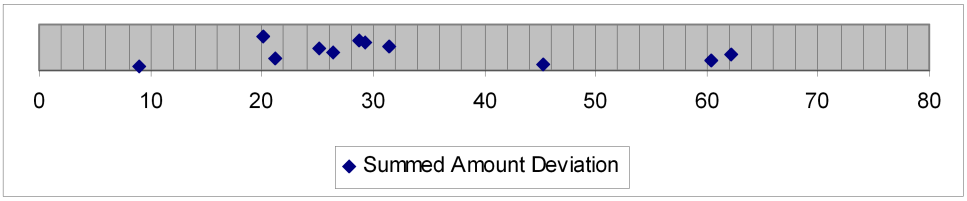


Figure 18: Distribution of Summed Absolute Deviations of Each Participant's Estimations (Palladio Technique, Design Option With Compression)

Technique	Average	Average Absolute Deviation	Variance	Standard Deviation
CB-SPE	43.4	9.7	179.1	13.4
Palladio	32.7	12.7	278	16.7

Table 18: Statistical Values of $devEst_p$ Distribution (Design Option With Compression)

For the design option with compression, the spread of input data is relatively high for both techniques, as can be seen in the figures 17 and 18. Note that the scale in this figure is different to the corresponding figures of the other design option, and that the deviation is much higher. This is due to the different compression rates the participants estimated.

Metric 2b, design option without compression The analysis of metric 2b is similar to the analysis of metric 2a. However, only one value per participant is considered, thus no sum

has to be calculated. For each participant, the $devPred_p$ is calculated like described in section 3.2.2:

$$devPred_p = |pred_p - avgPred|$$

with $pred_p$ being the result of participant p's performance prediction, i.e. the predicted response time, and $avgPred$ being the average predicted response time. Note that the result of a Palladio prediction is a distribution, not a single value. For this metric, the mean value of the distribution is considered, representing the predicted average response time. Table 19 shows the resulting absolute deviation of each participant's predicted response time to the average predicted response time. A graphical representation of the distribution of the predicted response time (for both CB-SPE and Palladio technique) are shown in figure 19 or 20, respectively. Descriptive statistical measures for this distribution are given in table 20.

CB-SPE											
Participant	1	2	3	4	5	6	7	8	9	10	
$devPred_p$	4.5	14.5	0.5	1.5	1.5	4.5	0.5	2.5	2.5	14.5	
Palladio											
Participant	1	2	3	4	5	6	7	8	9	10	11
$devPred_p$	12.2	36.2	35.2	50.8	16.5	56.1	77	48.9	41.1	56.9	44.5

Table 19: Absolute Deviation of Each Participant's Predicted Response Time (Design Option without Compression)

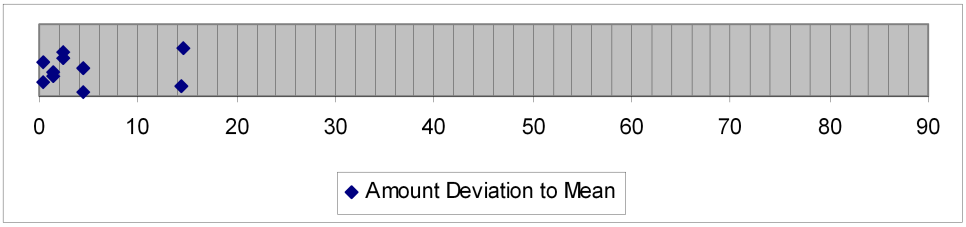


Figure 19: Distribution Of Absolute Deviation of Each Participant's Predicted Response Time (CB-SPE Technique, Design Option without Compression)

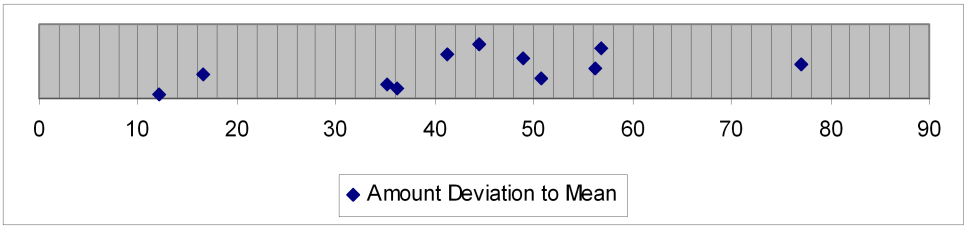


Figure 20: Distribution Of Absolute Deviation of Each Participant's Predicted Response Time (Palladio Technique, Design Option without Compression)

Technique	Average	Average Absolute Deviation	Variance	Standard Deviation
CB-SPE	4.7	3.9	28.6	5.3
Palladio	43.2	13.6	337.9	18.4

Table 20: Statistical Values of $devPred_p$ Distribution (Design Option Without Compression)

The statistical spread of the output of the CB-SPE technique is low. The average absolute deviation of the CB-SPE predictions (i.e. the mean of the distribution in figure 19) is even lower than that of the corresponding input. The deviation of the distribution is higher than that of the inputs, however, this difference is not high.

The statistical spread of the output of the Palladio technique is much higher than that of the CB-SPE technique. The mean value of the distribution in figure 20 is almost ten times that of the CB-SPE distribution. The deviation is higher, too. Thus, the statistical spread of the output is classified high.

Metric 2b, design option with compression Table 21 shows the resulting absolute deviation of each participant's predicted response time to the average predicted response time. A graphical representation of the distribution of the predicted response time (for both CB-SPE and Palladio technique) are shown in figure 21 or 22, respectively. Descriptive statistical measures for this distribution are given in table 22

CB-SPE											
Participant	1	2	3	4	5	6	7	8	9	10	
$devPred_p$	33.5	70.5	46.5	67.5	121.5	102.5	21.5	139.5	92.5	66.5	
Palladio											
Participant	1	2	3	4	5	6	7	8	9	10	11
$devPred_p$	36.3	114.1	142.2	2.5	114	34.6	21	64.5	3.4	5.1	0.5

Table 21: Absolute Deviation of Each Participant's Predicted Response Time (Design Option with Compression)

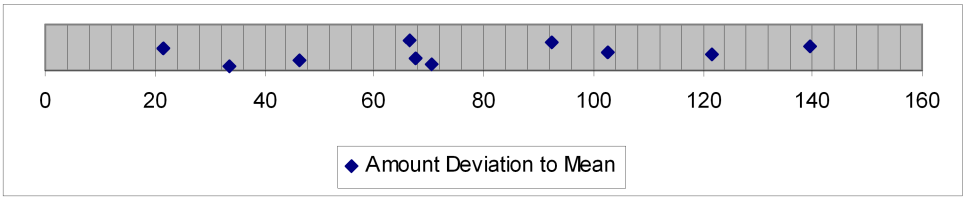


Figure 21: Distribution Of Absolute Deviation of Each Participant's Predicted Response Time (CB-SPE Technique, Design Option with Compression)

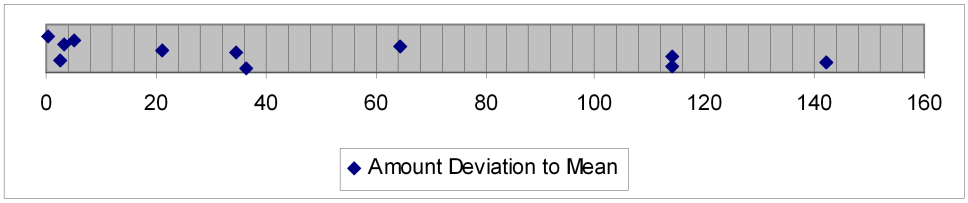


Figure 22: Distribution Of Absolute Deviation of Each Participant's Predicted Response Time (Palladio Technique, Design Option with Compression)

Technique	Average	Average Absolute Deviation	Variance	Standard Deviation
CB-SPE	76.2	30.2	1435.1	37.9
Palladio	48.9	43.5	2707.1	52

Table 22: Statistical Values of $devPred_p$ Distribution (Design Option With Compression)

For the design option with compression, the statistical spread of both output data is high, too, as can be seen in the figures 21 and 22. Here, like with the input values, this is due to the different compression rates the participants estimated. As the time in the network is a major part of the total response time, the compression rate greatly influences the response time.

As a result, to classify the performance prediction techniques, only the results for inputs and outputs of the design option without compression are used.

Classification The techniques are classified according to the matrix (table 2) presented for metric 2c. For the classification, only the results of the prediction for the design option without compression are used, as the results for the second design option are too heavily influenced by the compression rate. The statistical spread of the input data for the CB-SPE technique is low, and results in a low statistical spread of the output data. Thus, the CB-SPE technique is put in the forth cell of the matrix. The statistical spread of the input data for the Palladio technique is low, too (although slightly higher than that of the CB-SPE tool), its output, however, has a high statistical spread. Thus it is classified in the third cell, having a high sensitivity.

	Statistical spread of outputs: high	Statistical spread of outputs: low
Statistical spread of inputs: high	No statement about sensitivity is possible.	Sensitivity is low
Statistical spread of inputs: low	Sensitivity is high: Palladio	Sensitivity is alright: CB-SPE

Table 23: Metric 2c: Classification of Performance Prediction techniques

Note that the differences in the specified distributions for the Palladio prediction has not been taken into account here, only the estimated values are considered. The differences of the specified distributions is not as easy to assess as the other estimations.

5.2.2 Differences of Input Models

Due to the experimental setup, the input model of the CB-SPE technique only differ in the estimated values (cf. section 4.3.1). Because ArgoUML caused to many problems, I could not expect the students to create the required ArgoUML diagrams during the limited time of the experiment, thus, I prepared diagrams they only had to fill their estimations in. The difference of the input models cannot be assessed in this setting. The only differences, the estimations, do not depend on the used technique. Thus, an analysis of these differences do not help assessing the difficulty to create an input model.

For the Palladio technique, input models have been created by the participants, as they had to transfer the given sequence diagrams into service effect automata. Here, two main classes of resulting automata can be identified.

1. 5 out of 11 participants created a service effect automaton for each service (i.e. for DefaultHTTPrequestProcessorTools.SendContentToClient and ZipHTTPRequestProcessorTools.SendContentToClient), describing the internal computations and external calls executed when calling this service.

2. 6 out of 11 participants created only one service effect automaton for each design option. Their first service effect automaton describes the internal computations and external calls executed when calling DefaultHTTPrequestProcessorTools.SendContentToClient, too. Their second automaton describes the internal computations and external calls executed when calling DefaultHTTPrequestProcessorTools.SendContentToClient and ZipHTTPRequestProcessorTools.SendContentToClient in a row.

These differences do not affect the results, because the ”melting” of service effect automata is done during the calculations of the Palladio tool, too. However, the second class is not what a service effect automaton is supposed to be. Maybe the idea of service effect automata has not been perfectly understood by the participants. When drawing this conclusion, it has to be kept in mind that some of these 6 participants may have chosen this way fully aware of what they do, to simplify their later analysis.

The results show, however, that it is quite hard for the participants to find adequate distributions of their estimated values. A way to handle the distributions was given on the sheets: restricting the distribution as far as possible. A comparison with the measured distribution shows that this leads to a less spread distribution than measured. Some participants specified other distributions, many of them too even. However, they were not prepared to do do. Altogether, the preparation of the participants should have been better or the exercise should have proposed a better way to specify the distribution functions. However, should be delivered by preceding analyses, are not supposed to be estimated manually.

As the two variants of creating the input automaton do not affect the results, the difference of input automata is low. However, the specification of distribution functions was problematical.

5.2.3 Answers to Question 2: What influence do the input models have?

The results of metric 2c attest the CB-SPE technique a good applicability due to a moderate sensitivity. It is unknown how results are if high-spread inputs are used. However, the average absolute deviation of the results here is lower than the average absolute deviation of the estimations. Thus, one can suppose that the sensitivity is rather low, and that the technique could handle more wide-spread inputs, too. Of course, this is only speculation and should be supported by further experiments.

The results for the Palladio technique attest a high sensitivity, thus leading to a bad applicability. However, the influence of the different distributions used by the participants has not been taken into account, thus the classification may be wrong. Further experiments are needed to find an answer.

The difficulty to create input models could not be assessed for CB-SPE. The problems during the preparatory exercises, however, suggest that the creation of input models is problematically, at least at this point of time, using ArgoUML. However, these problems are merely technical and should not be used to assess the applicability of the technique itself.

The input automata for Palladio could be created by the participants. The differences in the input automata do not affect the results, and thus are rather stylistic. Thus, the input automata can be obviously extracted from given information. However, the specifying of distribution functions in this experiment are problematically. It has to be kept in mind that the participants had not been trained to do so. Thus, this flaw is likely to be a flaw of preparation and not of the technique.

5.3 Reasonableness of the Workload

5.3.1 Results of Intuitive Prediction

As mentioned in section 4.3.1, a comparison group has been asked to predict the performance of the system, too. Five computer scientists handed in solutions to the exercise. Metrics 1a and 1b have to be analyzed for their results. Figure 23 shows the predictions of the members of the intuitive group. In figure 24, the prediction for the design option with compression are compared to the function derived from the adjusted measured response time of the implementation.

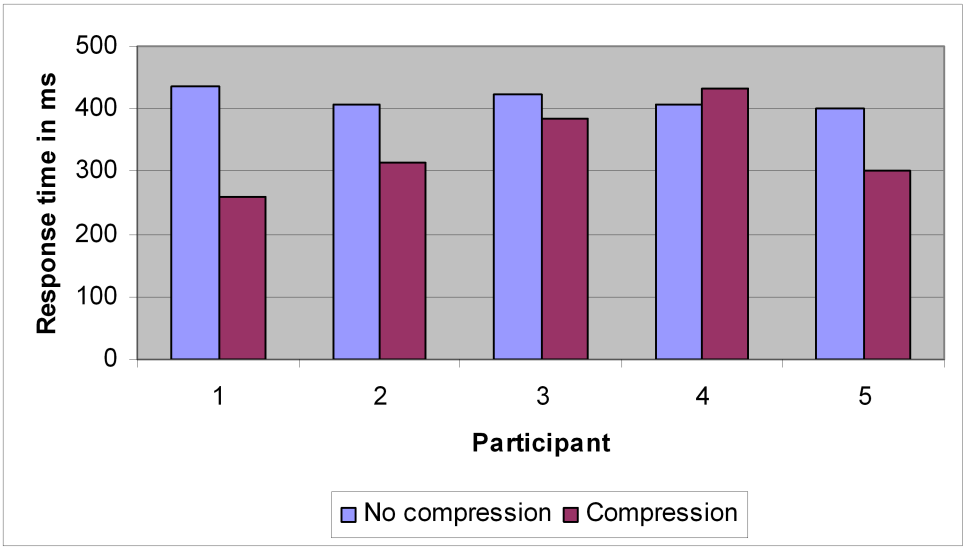


Figure 23: Intuitive Predicted Response Times for Web Server

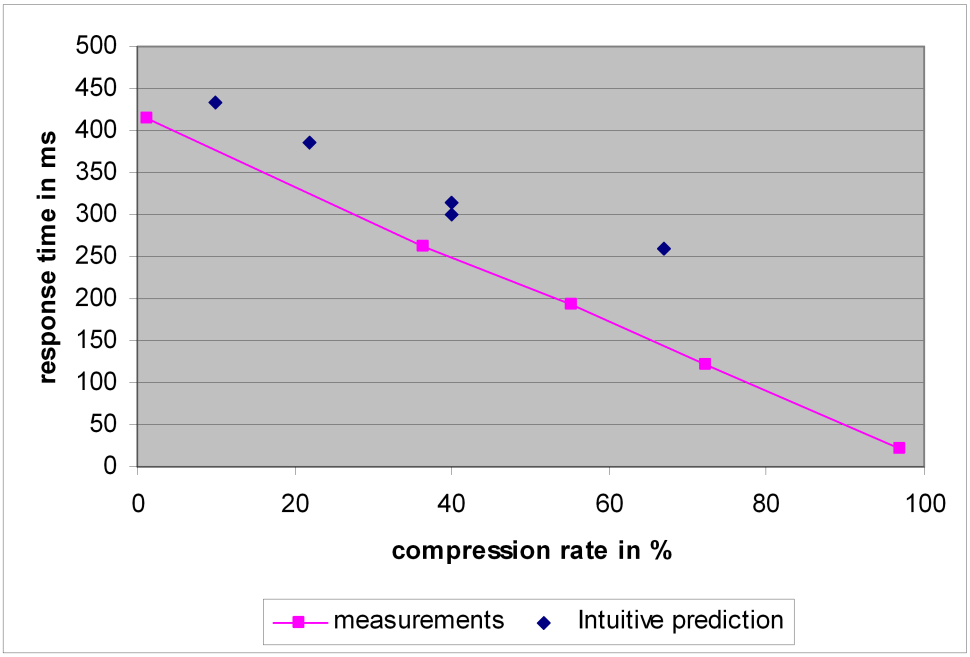


Figure 24: Comparison Adjusted Measured Response Times and Intuitively Predicted Response Times for Web Server with Compression

The predicted response times for the design option without compression range between 400 ms and 435 ms. The absolute deviation here from the adjusted measured average response time ranges from 0.16 ms to 22.84 ms. The average absolute deviation from the adjusted measured response time is 12.97 ms.

The predictions for the use of compression vary more, they range between 260 ms and 434 ms. Here, the absolute deviation from the function derived from the adjusted measured average

response time ranges from 51.9 ms to 116.5 ms. The average absolute deviation from the function is 71.1 ms.

Based on their predictions, 4 out of the 5 participants chose the correct design decision, the use of compression. Thus, the percentage of correct design decisions is 80%. As the size of the intuitive group is very small, this is probably no significant value. Additionally, the intuitive group was subject of the wrong expectation of the time consumption of the compression algorithm, too.

A further threat to the validity of the comparison group's results is their participation in the similar study of Koziolek [14]. In Koziolek's study, the design option with compression had the best response time, too. The members of the comparison group may have transferred their experiences with the before mentioned study into their predictions here. This threat cannot be eliminated, however, all solutions of members of the comparison group have discernible calculations, thus the members of the comparison group did not only rely on their former experiences.

5.3.2 Workload Metrics

Three metrics in this context capture times the participants needed to work on their tasks, namely metric 3c (the time needed to solve experiment exercise), metric 3d (the time to become acquainted with the techniques) and metric 3e (the time to solve the preparatory exercises). The participants were asked about the needed times anonymously. For both techniques as well as an intuitive prediction, the results for the metric 3c can be seen in table 26. For both performance prediction techniques, the results for metrics 3d and 3e are shown in table 25. Both tables can be found in the appendix. The average times for each technique as well as an intuitive prediction for the three metrics can be found in table 24.

Technique	Average Time to solve experimental exercise in minutes	Average time to learn technique in minutes	Average time to solve preparatory exercise in minutes
CB-SPE	122.3	141.8	363
Palladio	136.4	72.1	216
intuitive	49	0	0

Table 24: Average Needed Times

5.3.3 Correct Solutions and Corrections

23 solutions were handed in for the first preparatory exercise, 22 for the second. I only analyze the solutions handed in by participants of the experiment, as the other solutions have no significance here. In section 4.2.2, the results of the preparatory exercises are presented. To better analyze this metric, the results are repeated in this section, some enhancements needed for the interpretation of this metric are included.

The first exercise trained the CB-SPE technique. As mentioned in section 4.4.1, many experienced problems with the interplay of ArgoUML and the CB-SPE tool. Only 12 participants were able to analyze the system. Although no estimations were involved, the predictions of the participants varied. The reason for this variation is not always identifiable, some diagrams seem to be identical, but lead to different results. Thus, small variations are classified correct solutions. With this demand, 6 participants handed in correct solutions. The other participants with analyzable diagrams had small errors in their diagrams, e.g. they forgot a return transition, or did not model the control flow for alternatives and loops correctly. All these types of errors are due to a more complex exercise. The experimental exercise has a simpler control flow. The participants who could not analyze their diagrams mainly had syntactical errors.

The second exercise trained the Palladio technique. Here, 18 participants were able to analyze the system, three participants were unable to do so. One participant of those three had syntactical errors, for the other two the reason could not be found. 5 participant correctly modeled the system, although two of them had to comment out parts of their automaton to be able to do the analysis. The other participants did not model the automaton correctly. Here, as with the CB-SPE technique, the loops and alternatives were sources of errors.

For the experiment itself, only two participants using the CB-SPE technique were asked to correct their solutions. One participants left the experiment after little more than 2 hours, as he had other obligations, and finished the exercise later, thus having one correction. One participant seemed to have problems with the technique, especially with the use of same units for time and file size throughout the diagrams. After 5 corrections, he handed in a correct solution.

For the Palladio technique, 6 participants were asked to correct their solutions. Once, a participant had made a mistake when inserting the results of one analysis into a second service effect automaton. The other corrections concerned the specifying of the distributions of estimated values. As mentioned in section 4.4.2, its specification is error-prone.

5.3.4 Answers to Question 3: Is the workload of a prediction reasonable?

For this rather simple performance prediction, the intuitive prediction has advantages. The time needed for the intuitive prediction is much lower than for a prediction with a technique. One reason is that the comparison group was able to concentrate on the values that affect performance and add these to get the overall response time. They did not have to handle the techniques that are able to handle other tasks, too, and thus are more general.

Additionally, it has to be taken into account that the participants were not very experienced with the techniques. Although they received a beforehand training, the prediction during the experiment was only their second one. Thus, the time needed for a single prediction may further drop when the participants have gained more experience with the used techniques. The majority of participants was, however, prepared enough to handle the techniques. Only one participants for CB-SPE had to correct his solution. The corrections made for Palladio concerned the specification of a distribution for their estimation. This aspect of the Palladio technique was not enough well-trained. It may have lead to worse overall results for the Palladio prediction technique.

The percentage of correct design decisions was lowest in the results of the comparison group. However, as the group was small, the number is not very significant. Additionally, the wrong expectations of the time needed to compress a file influenced the one wrong decision. Anyway, it is shown that the predictions with techniques are at least as good as the intuitive predictions, if not better.

The time needed to learn the techniques and to work on the preparatory exercises are lower for the Palladio technique. Thus, the idea of service effect automata seems to be intuitive. However, the longer time for the CB-SPE technique may be a result of problems with ArgoUML, too. A different tool may improve the times needed to become acquainted with the CB-SPE technique.

For both techniques, more elaborated tools may lead to better times to become acquainted with the techniques. Of course, a tool that is more convenient to handle can be used faster. Additionally, more tasks for the user of the tools have to be automated, like deriving the service effect automata from sequence diagrams or inserting the results from the analysis of the performance of one service into the next. As both tools are developed for research purposes rather than practical applicability, there is much potential for improvement here.

For more complex tasks, the techniques are likely to be advantageous. When the complexity of the task becomes too great for an intuitive prediction to deal with, techniques can help. However, the tools used with the techniques have to be more convenient and have to automatize more. To prove this assumption, more complex system have to be analyzed within a further experiment or more alternatives have to be evaluated against each other.

6 Conclusions and Outlook

6.1 Validity of this Case Study

To assess the overall validity of this case study, its internal and external validity is looked at. The internal validity of an experiment describes how good disturbance variables are held constantly, i.e. how good influencing variables can be controlled. The external validity describes how good the results of the experiment can be transferred to other applications [18].

6.1.1 Internal Validity

Prechelt [18] defines the internal validity of a controlled experiment as follows. As I try to conduct this case study as close to a controlled experiment as possible, this definition is used.

> The internal validity of a controlled experiment is the degree to which the changes of the dependent variables were caused by only the changes of the independent variables, i.e. how good disturbance variables can be controlled.

The most important threats to internal validity are introduced in the following and their impact in this case study is analyzed.

Maturation: The maturation applies to changes of the participants behavior over the course of the experiment. There exist two main forms of maturation: On the one hand, a participants behavior changes because he or she learned during the experiment (Learning Effect) or applies conclusions from one part of the exercise to a later one (Sequence Effect). On the other hand, fatigue may change a participant's behavior.

In this experiment, each participant only has to solve one task, reducing the threat of learning and sequence effects. However, they had to analyze two design options successively. Here, sequence effects are theoretically possible. Looking at the design options more closely, however, it becomes visible that sequence effects have no impact here, as timing estimations that occurred in both design options were supposed to be identical. Fatigue effects are unlikely to occur during the two hours of the experiment.

Instrumentation: Not only the behavior of the participants, but the behavior of the experimenter or the arrangement of the experiment is subject to changes, invalidating the results of data collection. For example, subjective adjudgements issued experimenter may change during the process of judging.

In this case study, no subjective evaluations of the participants results have been made, as the results of the performance predictions are expressible in numbers. Their collection was tool supported. The tasks were issued on sheets and printed beforehand, thus the results of the first CB-SPE group did not influence the Palladio group's task.

History: Experiments that take several weeks can be subject to events that happen outside the experiment itself. For example, news in technical press referring to one of the tested or a similar techniques may change the participants attitude toward the techniques, thus influencing their motivation. Another simpler history effect are former participants giving away their results to prospective participants of the experiment.

In this case study, the experiment took place on one single day, thus events outside the experiment did not affect it. However, between the two groups, there was a break of two hours. Thus, the possibility that members of the two groups talked on the experiment during this time cannot be excluded. However, the proceeding of the two techniques differs sufficiently so that such a talk would not help the members of the second group. Additionally, no particular compliance between the estimation of the members of the first group and the members of the second group could be detected.

Selection: In principle, dividing the participants into groups should be random. However, circumstances can demand a certain division. For example, if there are insufficient participants, a random division could lead to a stronger and a weaker group. If the selection of the participants is not random, it has to be ensured that the selection does not affect the results of the experiment.

In this case study, the participants have been assigned to the groups. The goal of this division was to create equally capable groups, as described in section 4.2.2. No other selection criteria than the results of the preparatory exercises have been used, thus this selection lead to a more balanced group instead of resulting in seclection effects.

Regression: If a participant performs an exceptionally good (or exceptionally bad) performance for his or her means, it is likely that in a second test his or her performance will drop (or increase). If the participants have been assigned to groups based on pretests, this ”regression toward the mean” effect invalidates the selection.

As there were two preparatory exercises in this experiment, the probability of undetected regression effects is lower. Most participants have shown relatively constant performance in the two preparatory exercises. In the experiment session, no deviation from the participants formerly shown performance could be detected, too.

Mortality: Participants leaving the experiment during its conduction are a threat for internal validity, too. Other participants' motivation can be affected, additionally the selection of members for the groups is distorted.

In this experiment, two participants of the preparatory exercises did not take part in the experiment session. As the preparatory exercises were dealt with at home, their missing in the experiment session should have had no impact on the other participants. However, the two dropped-out participants had been considered when selecting the members of the groups. Due to their drop-out, the CB-SPE group was left weaker than before. However, as the CB-SPE group delivered good results in this study, the impact seems to be low.

Demand Characteristics: The way in which different techniques and exercises are presented to the participants can further threaten the experiments significance. For example, the experimenter may unknowingly improve his or her favorite test group's motivation by welcoming them more enthusiastic and friendly. Of course, such influences invalidate the results of the experiment.

In this case study, the involved experimenters presenting the techniques and accompanying the experiment were more closely related to the Palladio technique. However, they strove toward being neutral. The basis for the CB-SPE tutorial slides kindly have been sent in by Raffaela Mirandola, one of the developers of the CB-SPE technique. Thus, the introductions of the technique supposably can be compared. The exercises themselves were issued on sheets, only differing in parts that are different for the two techniques (c.f. exercise sheets in section B in the appendix).

Processing Errors: A further threat to internal validity is to erroneously measure the resulting data. Here, the errors range from wrong measurement instruments over mistypings when transferring measurement results to faults with applying statistical techniques.

In this experiment, the majority of the data has been collected by copy-pasting the participants results into spread sheets. Thus, mistyping errors are unlikely, except for the data of the questionnaire. The participants results were saved directly, so wrong measurements instruments do not apply here. Of course, the possibility of other faults in transferring the resulting data into the spread sheets cannot be entirely excluded.

6.1.2 External Validity

Prechelt [18] defines external validity as follows:

> The external validity of a controlled experiment is the degree to which its results can be transferred correctly to other applications - in particular those that frequently occur in practice. Aspects are, for example, motivation and qualification of participants, kind and size of the software, kind and form of the task as well as constraints like further software engineering methods, technical and spacious work environment, mental state, labor times, time pressure, quality requirements and the like.

To evaluate the external validity of this case study, I look at the aspects mentioned above more closely. Additionally, the size of the test groups is important for generalizability.

Motivation and Qualification of the Participants: As mentioned in section 4.1, the participants are advanced students, thus they were able to handle the fundamentals of computer science. Although the difference in qualification between advanced students and professionals is not very big according to Prechelt [18], an experiment with professional participants would have resolved the last doubts. The motivation of the participants was given by awarding marks. Depending on a students individual attitude, this may result in a lower or higher motivation. However, the motivation differs to the practical appliance of predicting the performance of a system that you developed yourself. However, this motivation can only be found in practice, not in experiments.

Size of the Test Groups: 10 participants applied the CB-SPE technique, 11 participants applied the Palladio technique. For a good generalizability, both groups should have had more members.

Kind and Size of the Software: The analyzed web server is a typical example for a systems whose performance properties are predicted. However, the size of the analyzed software is very small, compared to systems that have to be analyzed for their performance in practice. At this point of time, however, the techniques are not fully matured yet: To apply them in a large scale software project, the Palladio technique needs more automation to use the prediction of a services performance for another prediction. The presented problems with CB-SPE and ArgoUML prevents the use in large scale projects at this point of time, too. However, to assess the techniques potential, an empirical study of this size is appropriate.

A further point is that the measurement results depend not only on the web server design, but on the specific implementations. Thus, the exact measurements are not even transferable to different implementations.

Kind and Form of the Task: In connection with performance predictions, the possible tasks do not vary much. In this experiment, the participants are asked to predict a systems performance. Except for the size of the system (see above), the task is transferable to practical applications.

Further Constraints: There is no standard for the constraints presented in the definition above. For example, different organizations have different forms of these constraints. There have been no special constraints in this experiments that would hinder the transfer to other applications.

The results of this case study are likely not directly transferable to a practice appliance of the techniques. Although the student participants are not problematic, the test groups size is to low. Additionally, the analyzed system is indeed similar to systems in practice, but too small. However, as the practical application of the techniques is not given yet, the possibility of testing the techniques with large scale software projects is not given. Thus, the size and transferability of this experiment is adequate.

6.2 Summary

This case study shows the applicability of the two performance prediction techniques CB-SPE and Palladio for the prediction of the response time of a simple software architecture. In the experiment, 21 students used the two techniques to predict a web server's response time and choose a the design option out of two possible ones. Most participants (95%) choose the design option whose later implementation indeed had the best response time: The use of compression when sending a HTTP response to the client.

Altogether, the CB-SPE technique produced the better results. The absolute predicted response time for the design option without compression is closer to the measured one. Additionally, all CB-SPE participants chose the right design option. The predicted response times for the design option without compression are less true, however, here the information on the sheets are imprecise.

The predictions made using the Palladio technique are less right. Probably, this is due to the problem of specifying lower bounds for distribution function and the absence of information concerning the distribution of the called services and internal computations on the sheet. As no lower bounds can be specified, the predicted average response time is to much influenced by the outliers, as the average of a distribution is sensitive to outliers. A statistical metric that is less prone to outliers is the median, possibly a comparison of the techniques medians would lead to different results.

The sensitivity of the CB-SPE technique regarding changes in the input data is moderate, which results in a good applicability. To assess the behavior of the CB-SPE technique when confronted with wide spread inputs, further experiments have to take place. Taking only into account the estimated values, the Palladio technique has a high sensibility. However, the different distribution functions that have been specified by the participants have not been taken into account.

The implementation of both techniques is problematically, especially for CB-SPE. About half of the participants were not able to create input diagrams with the ArgoUML tool that are processable by the CB-SPE tool. Here, the implementation has to be improved. For the Palladio tool, the specification of a lower bound for distribution function misses. The use of both techniques could be more convenient, too, but it has to be kept in mind that they both are prototypical.

The time needed for a performance prediction by both test groups was higher than the time needed by the comparison group using no technique. At the same time, the predictions made with the help of the techniques have only slightly better results (here, it has to be kept in mind

that the comparison group was mislead by false information on the sheet). Thus, I conclude that for the simple system like analyzed here, the performance prediction techniques as they are now have no advantages. However, the techniques are not fully matured yet. A more mature technique would reduce the efforts of the developer, as more steps can be automatized. For the CB-SPE technique, it is planned to fully integrate the prediction into a UML modeling tool, by inserting the predictions results in the UML diagrams for immediate feedback. For the Palladio technique, the results of one services performance can be more straightforward included into a further analysis.

Altogether, the CB-SPE technique can be attested a good applicability when abstracting from the problems of the current implementation. To assess the Palladio technique, further experiments have to take place. In these experiments, more data must be present concerning the distribution functions or the participants have to be better trained to assess the distributions themselves. However, the experiment shows that the Palladio technique needs more effort to predict a performance of a system, as in addition to the estimations of plain timing values, a distribution has to be found. If this data is collected, however, the approach looks promising to actually derive extra information on a distribution function of performance values. Further experiments have to evaluate this statement.

6.3 Future Work

There are several points for further works in this area. The empirical comparison and validation could be conducted as a controlled experiment. Thus, its plausibility could be improved, as the observed results could be definitively traced back to the change of the experimental variables. However, it is even more important to look at the performance prediction of more complex systems or for more possible design options. In this study, the comparison group achieved similar results in less time. For more complex systems or more design options, a manual prediction may be too complex or even lead to wrong results, but this can only be evaluated by experimenting with a more complex system or more design options.

A third point of contact references the Palladio technique: Here, the potential of the specification of distribution functions has to be looked at. In this experiment, the possibility of specifying distribution functions was rather negative for the techniques results. It relates to the practical case in which no information about distribution is available. If so, the Palladio technique possibly should not be applied (or have reasonable default distributions). The case that the distributions are known, however, is not considered in this study and needs further investigation. Participants have to be trained.

In the context of this work, only the rightness of the techniques is looked. A further possibility is to evaluate the accuracy of the techniques, comprising both rightness and precision.

The duration of the measurement of the web server and thus the number of measurements have been arbitrary in this study. As more than 400 requests have been tested, the distribution function seems likely to be valid. However, the needed amount of measurements to get an appropriate distribution function may be determined, too.

List of Figures

List of Tables

References

[1] S. Balsamo, A. DiMarco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: A survey," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 295–310, May 2004.

[2] S. Balsamo, M. Marzolla, A. DiMarco, and P. Inverardi, "Experimenting different software architectures performance techniques: A case study," in *Proceedings of the Fourth International Workshop on Software and Performance*. ACM Press, 2004, pp. 115–119.

[3] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering - 2 Volume Set*, J. J. Marciniak, Ed., 1994, pp. 528–532.

[4] A. Bertolino and R. Mirandola, "Towards Component-Based Software Performance Engineering," in *Proc. 6th Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction, ACM/IEEE 25th International Conference on Software Engineering ICSE 2003*, 2003, pp. 1–6.

[5] ——, "CB-SPE Tool: Putting Component-Based Performance Engineering into Practice," in *Proc. 7th International Symposium on Component-Based Software Engineering, Lecture Notes in Computer Science*, vol. 3054, 2004, pp. 233–248.

[6] C. Differding, B. Hoisl, and C. M. Lott, "Technoloy package for the Goal Question Metric Paradigm," University of Kaiserslautern, AG Software Engineering, Tech. Rep., 1996.

[7] V. Firus and S. Becker, "Towards performance evaluation of component based software architectures," in *Proceedings of Formal Foundation of Embedded Software and Component-Based Software Architectures (FESCA), Electronic Notes in Theoretical Computer Science*, 2004.

[8] V. Firus, S. Becker, and J. Happe, "Parametric Performance Contracts for QML-specified Software Components," 2005, to be published in: *Proceedings of Formal Foundation of Embedded Software and Component-Based Software Architectures (FESCA)* and *Electronic Notes in Theoretical Computer Science*.

[9] M. Fowler, *UML Distilled*, 3rd ed. Addison-Weasley, 2003.

[10] S. Frølund and J. Koistinen, "QML: A Language for Quality of Service Specification," Hewlett Packard, Software Technology Laboratory, Tech. Rep. HPL-98-10, Feb. 1998.

[11] R. L. Glass, *Software Runaways: Lessons Learned from Massive Software Project Failures*. Prentice-Hall Publication, 1998.

[12] R. Guth and L. Radosevich, "IBM crosses the Olympic finish line," http://www.infoworld.com/cgi-bin/displayStory.pl?/features/980209olympics.htm, Feb. 1998, accessed July 21st, 2005.

[13] IC#Code, "SharpLibZip," http://www.icsharpcode.net/OpenSource/SharpZipLib/Default.aspx, accessed August 6th, 2005.

[14] H. Koziolek, “Empirische Bewertung von Performance-Analyseverfahren für Software-Architekturen,” Diplomarbeit, Carl von Ossietzky Universität Oldenburg, 2004.

[15] H. Koziolek and V. Firus, “Empirical Evaluation of Model-based Performance Prediction Methods in Software Development,” in *Software Architecture Quality and Software Quality*, ser. Lecture Notes in Computer Science, R. Reussner et al., Ed., vol. 3712. Springer Verlag, 2005, pp. 188–202.

[16] Microsoft Corporation, “Guidelines For Providing Multimedia Timer Support,” www.microsoft.com/whdc/system/CEC/mm-timer.mspx, 9 2002, accessed July 22nd, 2005.

[17] R. Montealegre and M. Keil, “De-escalating Information Technology Projects: Lessons from the Denver International Airport,” *MIS Quarterly*, vol. 3, pp. 417–447, 2000.

[18] L. Prechelt, *Kontrollierte Experimente in der Softwaretechnik*. Springer Verlag, 2001.

[19] Red Gate Software, “ANTS Profiler,” http://www.red-gate.com/code_profiling.htm, accessed and downloaded June 18th, 2005.

[20] R. H. Reussner and H. W. Schmidt, “Using parameterised contracts to predict properties of component based software architectures,” in *Workshop On Component-Based Software Engineering (in association with 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems), Lund, Sweden, 2002*, I. Crnkovic, S. Larsson, and J. Stafford, Eds., 4 2002.

[21] R. H. Reussner, H. W. Schmidt, and I. H. Poernomo, “Reasoning on Software Architectures with Contractually Specified Components,” in *Component-Based Software Quality: Methods and Techniques*, A. Cechich, M. Piattini, and A. Vallecillo, Eds. Springer-Verlag, Berlin, Germany, 2003.

[22] A. Schmietendorf and A. Scholz, “Aspects of Performance Engineering - an Overview,” in *Performance Engineering: State of the art and current trends, Lecture Notes on Computer Science*, R. Dumke, Ed., vol. 2047. Springer Verlag, 2001.

[23] M. Sitaraman, G. Kulczycki, J. Krone, W. F. Ogden, and A. L. N. Reddy, “Performance specification of software components,” in *SSR ’01: Proceedings of the 2001 symposium on Software reusability*. New York, NY, USA: ACM Press, 2001, pp. 3–10.

[24] C. U. Smith, *Performance Solutions: A Practical Guide To Creating Responsive, Scalable Software*. Addison-Wesley, 2002.

[25] L. Tolke, “Frequently asked questions for ArgoUML,” http://argouml.tigris.org/faqs/users.html, accessed April 25th, 2005.

[26] Web Performance Inc., “Web Performance Trainer,” http://www.webperformanceinc.com, accessed and downloaded June 17th, 2005.

[27] X. Wu, D. McMullan, and M. Woodside, “Component Based Performance Prediction,” in *6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*, 2003.

A Preparatory Exercises and Tutorial Slides

A.1 CB-SPE Tutorial Slides

Empirical Comparison of the Model-Driven Performance Prediction Techniques CB-SPE and Palladio

Raffaela Mirandola
Università di Roma „Tor Vergata“
Viktoria Firus, Anne Martens
Universität Oldenburg

Organisatorisches (1)

- 31.05. 2005
  - 1. Tutorium CB-SPE
  - Ausgabe 1. Übungszettel
- 07.06. 2005
  - 2. Tutorium Palladio
  - Abgabe 1. Übungszettel
  - Ausgabe 2. Übungszettel
- 14.06. 2005
  - Abgabe 2. Übungszettel

Organisatorisches (2)

- 21.06. 2005
  - 3. Tutorium: Besprechung der Abgaben
- 28.06. 2005
  - Experiment
- Abgaben der Lösungen per Mail (vgl. Übungszettel)

Towards Component-Based Software Performance Engineering

Raffaela Mirandola

Dipartimento di Informatica, Sistemi e Produzione
Università di Roma Tor Vergata, Roma, Italy
mirandola@info.uniroma2.it

Outline

- Introduction:
  - from the software model to the validation of Extra-Functional Requirements (E-FRs)
- Performance validation
  - SPE
- Component-based systems
- CB-SPE

Early Software Validation

Evaluating
in the upper lifecycle phases
whether the software *model*
does
has the ability of doing
what the customer wants

… what the customer wants is (more or less completely and correctly) specified in the software requirements

Software requirements

- Functional : statements of services the software system should provide, how it should react to particular inputs and behave in particular situations
- Extra-functional : constraints on the services offered by the software system affecting the software quality

Software validation of E-FR

It is a common practice to validate a software model mostly against functional requirements rather than against extra-functional ones

Motivations from outer world

• Different (and often not available) skills required for E-FR modeling and validation

• Short time to market, i.e. quickly available software products performing quite poorly "seem" to be more attractive nowadays!

"… then why to pursue early validation of E-FRs?"

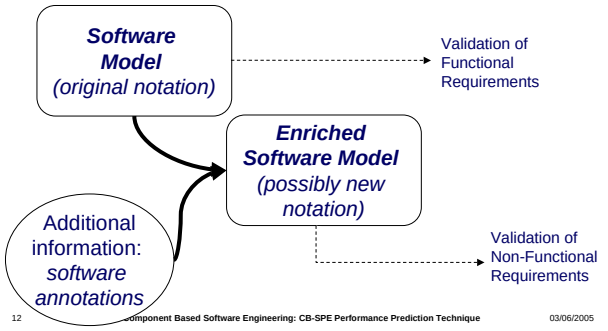
- In the early phases of the lifecycle a validation of E-FRs may prevent late inconsistencies hard to fix
- E-FRs are more critical in modern (possibly distributed) component-based software systems

"What is the gap between software development and E-FR validation?"

1. Amount and type of missing info to integrate into a basic software model in order to perform a E-FR validation
2. Algorithms to make automatic the step:

basic software model + additional info → *"ready-to-validation" model*

A general validation scheme



PERFORMANCE VALIDATION

Performance

"Theoretical" performance definition:

Time behavior (ISO 9126):

The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions

Performance metrics

"Practical" examples:

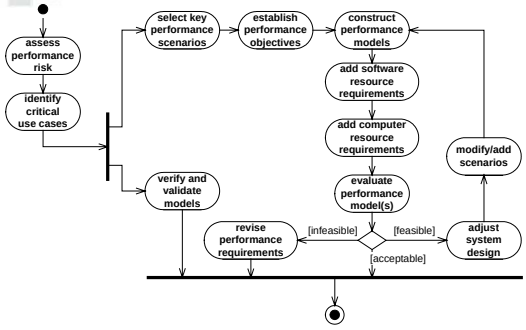
- *Response time*
- *Throughput*
- *Device utilization*
- *Scalability*
- *Reaction time (in real time systems)*

Software Performance Engineering

Software Performance Engineering

- Systematic quantitative approach to constructing software systems that meet performance objectives.
- Based on the methodical assessment of performance issues from requirements to implementation.

SPE Process



SPE Process (2)

1. Assess Performance Risk: the level of risk and its impact on system performance determines the amount of effort to put into SPE activity

2. Identify Critical Use Cases, i.e., those use cases that are mostly important to responsiveness or scalability for the user(s) of the system

3. Select Key Performance Scenarios, i.e., those that are executed frequently or that are perceived as critical to the performance

4. Establish Performance Objectives, i.e., for each key performance scenario specify quantitative criteria for evaluating its performance characteristics and for each combination of scenario and performance objective specify the conditions (workload mix and intensity) under which the performance objective should be achieved

SPE Process (3)

5. Construct Performance Models

separates the Software Model (SM)
from its environment/machinery model (MM)

- allows for defining software and machinery models separately and solving their combination,
- improves the portability of models

6. Determine Software Resource Requirements, i.e., the amount of processing and software resources required for each scenario step.

SPE Process (3)

7. Add Computer Resource Requirements, i.e., the load imposed on the devices used by scenario steps. Computer resource requirements depend on the environment in which the software executes

8. Evaluate Performance Models: using the model and the selected analysis method, compute the performance predictions. If feasible: choose the most promising design approach; otherwise, if infeasible, change product requirements

9. Verify and Validate the Model: these activities proceed in parallel with the construction and evaluation of the models. Model verification, for example, determines if the estimated resource requirements are reasonable. Model validation ensures that we are building a model that accurately (faithfully) reflects the target system.

Software Model

- characterizes the resource requirements of the proposed software alone, in the absence of
  - other workloads
  - multiple users
  - delays due to contention for resources

Software Model

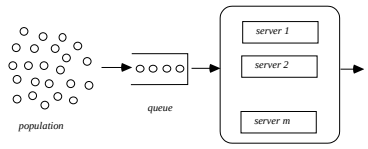
- provides static analysis of
  - best-case
  - worst-case
  - average

  response times
- is generally sufficient for identifying serious performance problems at the architectural and early design phase

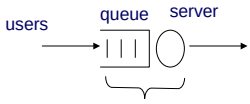
Machinery Model

- represents the key computer system resources as queues and servers.
  - a server represents a component of the environment that provides some service to the software (e.g., processor, disk)
  - a queue represents jobs waiting for service

Queueing Model



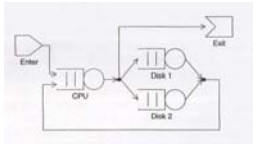
Queueing Network Models



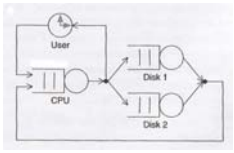
- Components
- Topology
- Parameters
  - job classes
  - job routing among centers
  - scheduling discipline at service centers
  - service demand at service centers

Queueing Networks

- open QN:
  - incoming and outgoing jobs to the network
  - Number of jobs varies over the time



- closed QN:
  - no external arrivals to the network
  - circulation of fix number of jobs



(Folie von Heiko Koziolek)

Component-based systems

CBSE

Rapid assembly of systems from components where

- components and frameworks have **certified properties**
- these certified properties provide the basis for **predicting** the **properties** of systems built from components [SEI00]

Properties
- Functional → correct component
- Extra-functional (performance, reliability) → useful component

Motivation

Performance specifications are essential for two basic reasons:

1. Multiple implementations provide the same functional behavior.

   Components that best fit the client performance requirements.

   Software components should include descriptions of performance behavior

2. If components have performance specifications, then the performance of the system can be derived based on the components it directly uses; the component implementations need not be re-analyzed in each new context they are used.

Performance (1)

Metrics such as application and component execution times, response times, resource utilization

Depends on characteristics of the systems such as:

- Choice of the architecture
- Choice of the programming paradigm
- Underlying platform resources
- Parallelism
- Distribution
- ...

Performance of component-based systems

- Source code not available
- Possible dependance on external components
- Possibly distributed application
- Heterogeneity in the underlying machine configurations

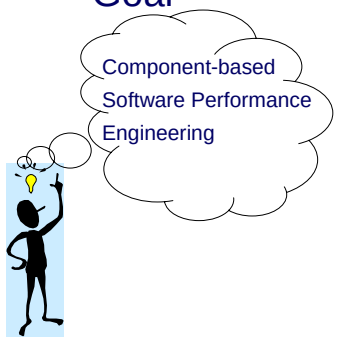
What to do?

Measurement

- Platform specific
- code (not) available
- costly

➡ Basing on models

Goal



... some pieces are already round here ...

- SPE
- UML

UML

Specification of component based on UML

UML cannot express several important (extra functional) system requirements such as: response time, availability, throughput and bandwidth.

Performance Analysis profile:
RT-UML standard

RT-UML(1)

- The RT UML Profile defines a standard set of stereotypes for expressing **platform-related** concepts:
  - resources
  - concurrency mechanisms
  - time and timing mechanisms
- The PA sub-profile:
  - Allows accurate specification of key performance concepts directly on UML models
  - Eliminates the need for manual construction of a separate performance model (i.e., the model can be derived automatically)

RT-UML(2)

PRIMA-UML, Propean, …..

Automatic (!!) generation
of queueing network based performance models
at different abstraction levels and
their parametric analysis

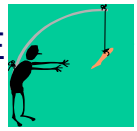
Component-based SPE

- Developed by Antonia Bertolino (1) and Raffaela Mirandola (2)

- *Towards Performance Based Software Performance Engineering.* Proc. 6th Workshop on Component-Based Software Engineering (CBSE), 2003
- *CB-SPE Tool: Putting Component-Based Performance Engineering into Practice*. Proc. 7th International Symposium on CBSE, 2004

- (1) I.S.T.I., Italian Research Council
- (2) Università di Roma „Tor Vergata“

Component-based SPE

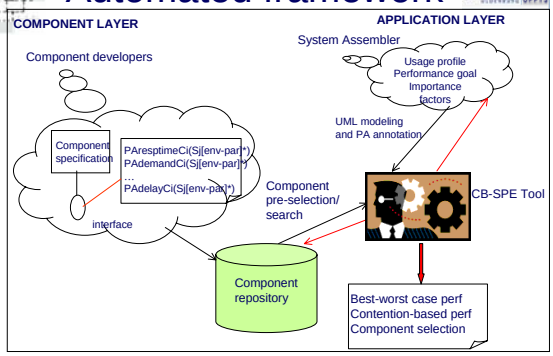


Two usage layers:

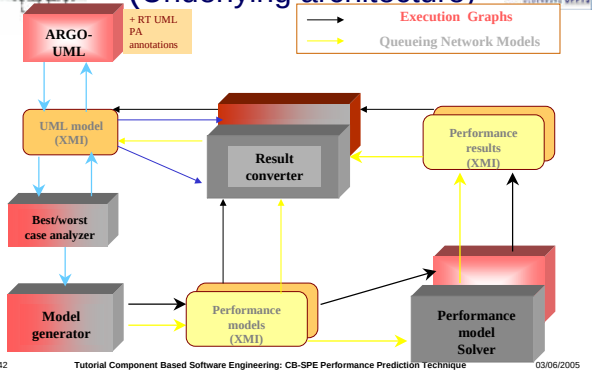
The component layer: guarantees to have components with certified performance properties (*component developer*)

The application layer: guarantees to have CB applications with the required performance *(system assembler)*

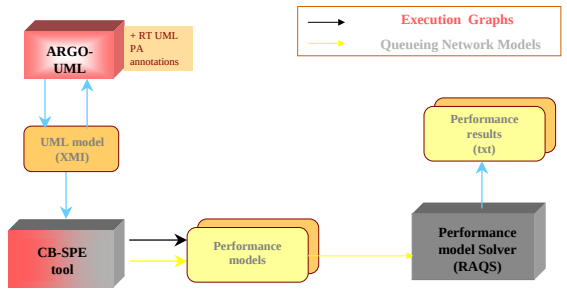
CB-SPE: Automated framework



CB-SPE Tool (Underlying architecture)



CB-SPE Tool (Underlying architecture)



CB-SPE Process

- Input: set of components with performance parametric annotations on the functionality interfaces (component developer)
- Output: selection of the components and modeling of the application with performance requirements satisfied or otherwise declaration of performance requirements unfeasibility

CB-SPE Process

- Step 1. Determine the usage profile (system assembler)
- Step 2. Component selection (system assembler)
- Step 3. Modeling and annotation (system assembler)
- Step 4. Best case analysis (automatic)
- Step 5. CB-SPE model generation (automatic)
- Step 6. Model evaluation (automatic)
- Step 7. Analysis of results (system assembler)

CB-SPE tool: a proof of concept

Strength points:

specification → performance prediction

underlying sound methodology (SPE)

automatic, compositional, hierarchical

Weakness points:

component specification

usage profile (input domain analysis)

...

Challenges (1)

New tool including CB-SPE

ECLIPSE

Profile definition

mobility

performance

automatic performance model generation

[Work in progress]

Challenges (2)

Web services

composition

model “on the fly”

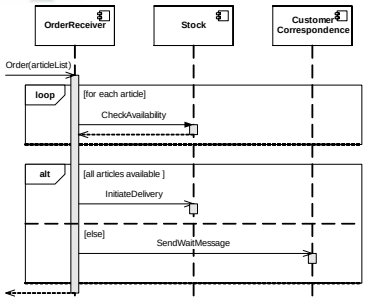
Best/worst case analysis

Contention based analysis

[Work in progress]

Use CB-SPE

Example Service



Want to know:

CPU elapsed time
Communication delay

Additional information

- Control flow:
  - Assume mean amount of articles to be 10.
  - Assume probability of all articles being available is 0.4
- Number of users: 100

Deployment

- OrderReceiver and Customer-Correspondence are deployed on a server called localServer.
- Stock is deployed on another server called warehouseServer.
- The two servers are connected with a fast leased line connection.

Metrics

- Processing time:
  - Demand: Number of work units
  - Throughput: How many work units per time unit
  - → Goal: Time consumption in time unit
- Communication delay:
  - Demand: Message size
  - Throughput: How many messages of size 1 per time unit
  - → Goal: Time consumption in time unit
- Use same time unit!

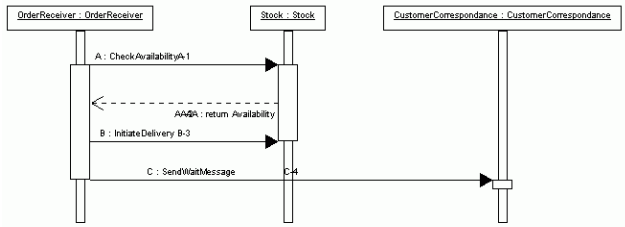
Processing Values

- CPU demand
  - OrderReceiver needs 1000 work units to call CheckAvalability, 2000 work units to call SendWaitMessage and 3000 work units to call InitiateDelivery
  - Stock needs 3000 work units to check the availability of an article.
- Throughput:
  - localServer: 30‘000 work units per sec.
  - warehouseServer: 10‘000 work unit per sec.

Communication Values

- Communication delay:
  - Message size measured in bit
  - Time unit is sec
- We assume:
  - Used leased line's speed is 10 Mbps
  - Size of messages:
    - Checking for availability: 130 Kbyte = 1.04 Mbit
    - Returning availability t/f: 100 Kbyte = 0.8 Mbit
    - Initiating delivery: 250 Kbyte = 2 Mbit

Resulting ArgoUML Sequence Diagram



Resulting ArgoUML Sequence Diagram

- PA Annotations:

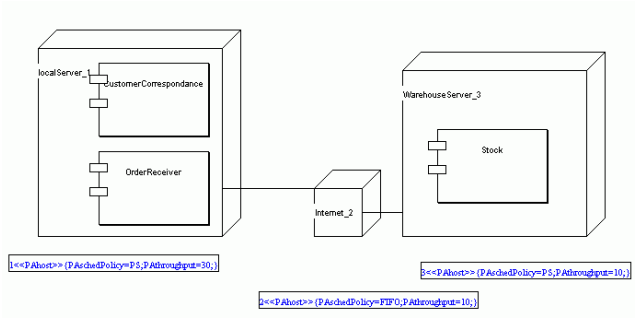
1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);PAdemand=(est,mean,1);PAsd=1;}<<PAclosedLoad>>{PApopulation=100;PAextDelay=1;}

2<<PAstep>>{PArep;PAextOP=(Internet_2,0.8);PAdemand=(est,mean,3;}

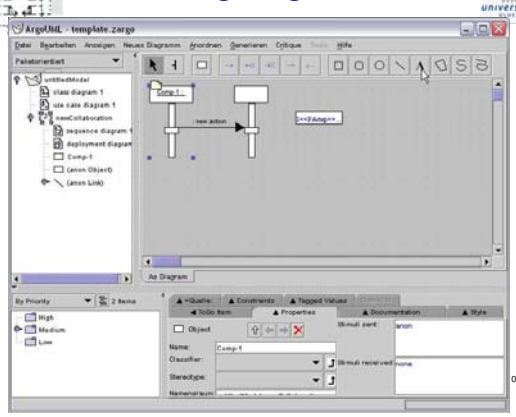
3<<PAstep>>{PAextOP=(Internet_2,2);PAdemand=(est,mean,3);PAprob=0.4;}

4<<PAstep>>{PAdemand=(est,mean,2);PAprob=0.6;}

Resulting ArgoUML Deployment Diagram



Using ArgoUML

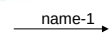


PA Annotations

- RT UML PA sub-profile
- Stereotypes <<PAxx>>
  - Followed by tags in braces
  - Tags are separated with semicolons

<<PAxx>>{PAtag1=value;PAtag2=value;}

PA Annotations in SD



1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);}

- Name of calls must end with a hyphen (-) and an ascending number, to associate the node with a RT-UML annotation.
- Each RT-UML annotation in the SD begins with the number of the call it belongs to.
- No number must be omitted (e.g. do not name three nodes x-1, x-2, x-8)

PA Annotations for SD

1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);PAdemand=(est,mean,1);PAsd=1;}<<PAclosedLoad>>{PApopulation=100;PAextDelay=1;}

- <<PAclosedLoad>>
  - Once in diagram, associated with first call
  - Defines PApopulation and PAextDelay
- <<PAstep>>
  - For each call
  - Defines PAextOP, PAdemand and control flow annotations

<<PAclosedLoad>>

1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);PAdemand=(est,mean,1);PAsd=1;}<<PAclosedLoad>>{PApopulation=100;PAextDelay=1;}

- Must be inserted in the first note of the SD.
- QN is closed (cf. slide 27)
- PApopulation;
  - The size of the workload (number of system users)
- PAextDelay
  - Tag must be defined and have a value
  - No further effect

<<PAstep>>

4<<PAstep>>{PAdemand=(est,mean,2);PAprob=0.6;}

- PAdemand
  - Used to model the demand to host on which the sending component is deployed.
  - est,mean for estimated, mean. Has no effect.
- PAextOP
  - optional
  - Used to model the service demand to resources like network
  - and so to model the communication delay
  - PAextOP=(resourceNode,messageSize);

3<<PAstep>>{PAextOP=(Internet_2,2);PAdemand=(est,mean,3);}

<<PAstep>>

- PAprob
  - The probability to execute this step.
  - A sequence of steps must have a PAprob sum of 1.

4<<PAstep>>{PAdemand=(est,mean,2);PAprob=0.6;}

- PArep

1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);PAdemand=(est,mean,1);}

2<<PAstep>>{\PArep;PAextOP=(Internet_2,0.8);PAdemand=(est,mean,3);}

  - To model a loop
  - Defines the number of repetitions
  - End tag \PArep in a following or the same step to mark end of repeated block.

Note:

- An optional call can only be modeled by providing a dummy alternative having no execution time and no extOP, to have probabilities‘ sum of one.
- Only one statement can be addressed by a probability. Branches are not possible.
- PArep and PAprob don‘t fit, as PAprobs must have sum of 1.
  - To model a loop containing two alternatives, use a dummy call to start and end loop.

<<PAstep>>

1<<PAstep>>{PArep=10;PAextOP=(Internet_2,1.04);PAdemand=(est,mean,1);PAsd=1;}<<PAclosedLoad>>{PApopulation=100;PAextDelay=1;}

- PAsd
  - Not RT-UML
  - Introduced to model the probability of a single SD to be executed
  - Must be inserted in the first note of the SD.
  - The sum of the probabilities of SDs must be equal to 1.

PA annotations for DD

- Name of nodes must end with an underscore (_) and an ascending number, to associate the node with a RT UML annotation.
- Each RT-UML annotation in the DD begins with the number of the node it belongs to.
- No number must be omitted (e.g. do not name three nodes _1, _2, _8)

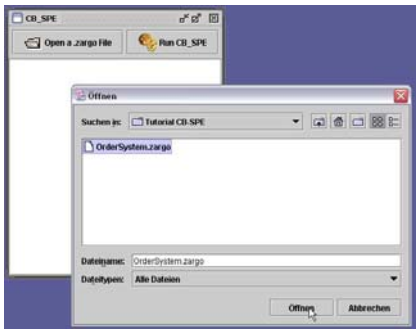
<<PAhost>>

1<<PAhost>>{PAschedPolicy=PS;PAthroughput=30;}

- Preceded by number of associated node
- PAschedPolicy
  - Tag must be present and have a value
  - Has no further effect
- PAthroughput
  - Defines the speed of the device

Feed CB-SPE Tool

- Annotated sequence diagram composed with ArgoUML

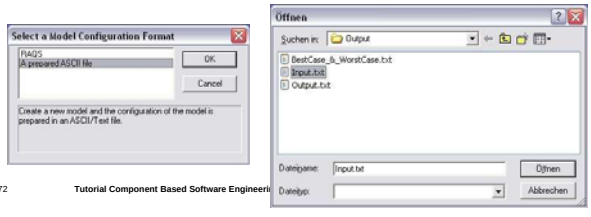


Run CB-SPE

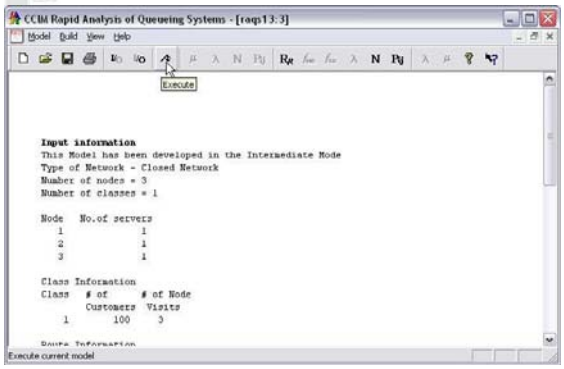
- Computes
  - Best/worst case
  - Execution Graph
  - Queueing Network

Feed RAQS

- Rapid Analysis of Queueing Systems
- Generated Input.txt contains QN
  - EG data included



Execute RAQS



Analyze Results

- Average time consumption: 20.003 sec
- Bottleneck node 3: Warehouse Server

```
Output Report
This Model has been developed in the Intermediate Mode
Type of Network = Closed Network

Network Measures
Average Time in the Network =  20.003

Node Measures
Node Util   ThruPut  AvTIQ   AvNIQ   AvTAN   AvNAN
1    0.167  4.99936  0.007   0.033   0.040   0.200
2    0.520  4.99936  0.110   0.552   0.214   1.071
3    1.000  4.99936  19.548  97.729  19.748  98.729

Class Specific Output
Class   AvRT      Thruput
    1   20.003    4.999

Util - the Utilization at a Node
Thruput - Output rate at a node
AvTIQ - Average waiting time in queue at a node
AvNIQ - Mean queue length at a node
AvTAN - Average time spent at a node
AvNAN - Mean number of customers at a node
AvRT - Average time spent in the network by a customer in a class
```


Conclusions

- Compare results with performance requirements
- If performance requirements are not fulfilled:
  - Try other adequate components, if available
  - Else revise requirements

Tipps

- Die Werkzeuge gibt es im Stud.IP
- Das CB-SPE Werkzeug muss im Verzeichnis C:\CB_SPE liegen
  - Entweder Install.bat benutzen
  - Oder es dorthin kopieren und Unterverzeichnisse Input und Output erstellen
- RAQS und ArgoUML einfach entzippen

Tipps

- Das Deployment- und Sequenzdiagramm muss den richtigen Namen haben. Am besten das template.zargo Projekt benutzen.
- ArgoUML keine Undo Funktion: Öfter Zwischenversionen unter anderem Namen speichern.
- Vgl. README.txt
- Falls es Fragen gibt: anne.martens@informatik.uni-oldenburg.de

A.2 CB-SPE Preparatory Exercise

Übungsblatt 1

Evaluation of Model-Driven Performance Prediction Techniques: CB-SPE Technique

Please work on the exercises alone. Send solutions to Anne until June, 7th 2005.

Exercise 1:

- Install the CB-SPE tools (can be downloaded from Stud.IP)
  - ArgoUML: Version 0.12, to be able to draw sequence diagrams.
  - CB-SPE Must be installed in C:\CB_SPE as the program contains hard coded paths. Create subfolders Input and Output or use Install.bat, which copies the files to C:\CB_SPE and creates the subfolders.
  - RAQS: Open c:\CB_SPE\Output\input.txt to analyze CB-SPE's output.
- Make yourself familiar with their functions by analyzing the case study example. For further reading, two papers by A. Bertolino and R. Mirandola presenting the technique are available in the Stud.IP portal.

Help using the CB-SPE tool:
- Be sure that calls and nodes are named correctly (-1, -2, ... resp. _1, _2, ...)
- Add call resp. node number to PA annotation.

Exercise 2:

The following sequence diagram (see appendix) models a system that finds flights for a user. The user specifies the time (including the date) and the start and destination airports. The FlightFinder component first requests all available AirlineWebServices from a service brocker. Afterwards, it requests general information needed for communication from the web services, if they are not already known.
After obtaining the general information, the available flights are requested from each AirlineWebservice. This happens successively. If there are flights available from one of the requested Airlines, the details for the cheapest flight are requested from the corresponding airline. Before requesting the details of the cheapest flight, however, the flights have to be ordered by price.

All components are located on different servers, that are connected by internet. For simplicity, model the different AirlineWebServices as one component on one server. Note that this does normally not allow an analysis with a population higher than 1.

Additional control flow information
- The chance that an AirlineWebService is not known to the FlightFinder is 10 percent.
- The chance that no flights are available is quite low, being 1 percent.
- Assume that the ServiceBroker returns 10 AirlineWebServices, i.e. GetFlights is called ten times.

Load information:
- Before each call to the other components, the FlightFinder component has a CPU demand of 2 work units, except before requesting the details of the cheapest flight, where the CPU demand is 50 work units. This is caused by the sorting of the flights by price.
- To return the available webservices, the service broker component has a CPU demand of 10 work units.
- To return its general information on a GetAirlineInformation call, the AirlineWebServices have a CPU demand of 3 work units. To find the flights for a given time and route, the AirlineWebServices have a CPU demand of 200 work units. Finally, to get the details of a flight, an AirlineWebService has a CPU demand of 5 work units.

Message sizes:

Call	Message Size in Byte
GetAirlineWebServices	4
Return AirlineWebServices	80
GetAirlineInformaton	4
Return AirlineInformation	10
GetFlights	10
Return Flights	80
GetDetails	5
Return Details	20

Remember to convert all bytes to bit or the other way round.

Deployment information:
Internet: 8 mbit/s connection.
FlightFinder's server: Throughput of 15 work units per msec.
ServiceBroker's server: Throughput of 65 work units per msec.
AirlineWebservices' server: Throughput of 65 work units per msec.

Your tasks:
Other designers think about deploying the FlightFinder on a faster, but more expensive server. You are supposed to find out whether this would increase the performance significantly or whether there are other bottlenecks.

a) Analyze the webserver component's performance using the CB-SPE technique. The metrics of interest are CPU elapsed time and communication delay. Assume that only one user uses the system.
b) Assume that all AirlineWebService components are deployed on one single server. With that, analyze the performance for 10 users using the system concurrently.
c) Propose actions to improve performance.

Send your solutions, i.e. the zargo project, the RAQS output, proposed actions to improve performance and an explanation of your decision to anne.martens@informatik.uni-oldenburg.de. Please include "CB-SPE exercise" in the subject and your name.

If your ArgoUML project cannot be processed by the CB-SPE tool, compare it with the case study projects downloaded with the CB-SPE tool an check your notations. If there are any further questions, don't hesitate to ask Anne.

Tip:
Only single calls can be annotated with a probability. However, always two calls are executed together in this example. To model these calls, annotate each single call with the corresponding probability.

Thus, model:

With a probability of x A and B are executed.

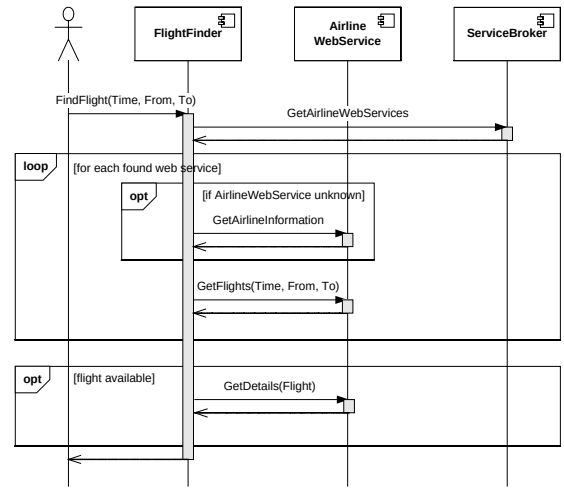
as

With a probability of x A is executed.
With a probability of x B is executed.

This does not model the control flow correctly, but the changes are minor. A new Eclipse based version of CB-SPE will be able to handle these cases correctly.

APPENDIX

Sequence diagram:



A.3 Palladio Tutorial Slides

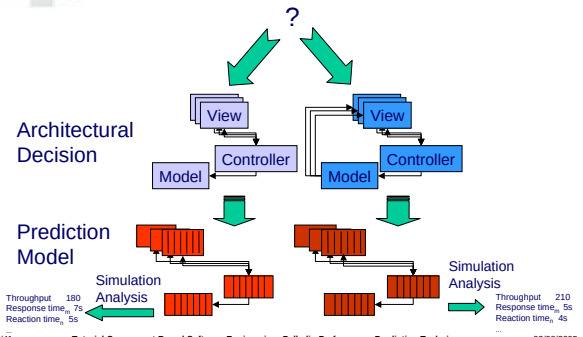
Performance Prediction Technique: Palladio

CARL VON OSSIETZKY universität OLDENBURG
Deutsche Forschungsgemeinschaft DFG
OFFIS
OLDENBURGER FORSCHUNGS- UND ENTWICKLUNGSINSTITUT FÜR INFORMATIK-WERKZEUGE UND -SYSTEME

Overview

- Motivation
- Approach of the Palladio Technique
- Modelling
  - Enhanced Service Effect Automata
- Using the Tool

Early Evaluation of Software-Architectures



Component Performance

- Component's performance results from
  - Performance of called services
  - Performance of underlying code
  - Hardware, middleware, ...
  - Operational profile

Palladio Technique

- Palladio Research Group

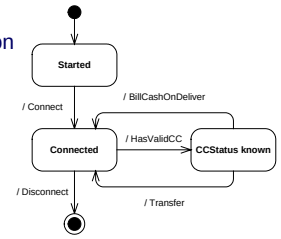
- V. Firus, S. Becker und J. Happe, *Parametric Performance Contracts for QML-specified Software Components*, 2005, to be published in: Electronic Notes in Theoretical Computer Science.

Approach

- Modelling the response time of a single service of a component in relation to the response times of the called external services
- No fixed response times to model influences of environment (hardware, internal state, amount of data, ...)

Service Effect Automaton

- Service Effect Automata
  - Finite State Machines
  - Control-flow abstraction
- Transitions correspond to external calls
- Nodes correspond to internal computations

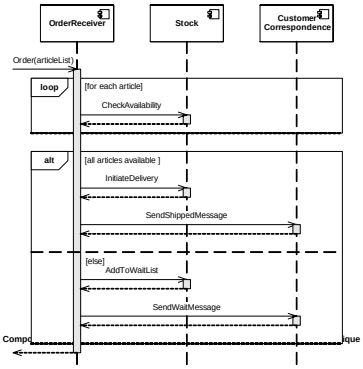


[1]

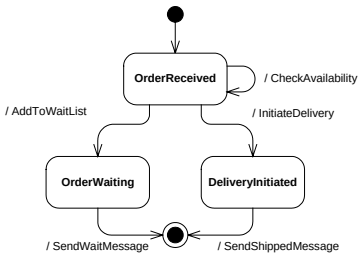
Creating a SEA for a service

- Start with an initial State
- Add transition for each external call
- Model control flow with loops and alternatives

Example Service



Example: Resulting SEA

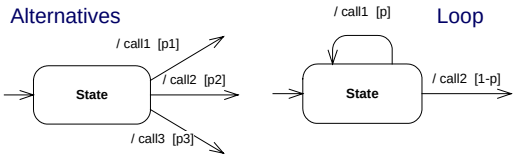


Additional Annotation

- Additional values needed for performance prediction
- Probabilities for different control flows
  - Probability to exit a loop
  - Probabilities for different alternatives
- Time consumption of nodes and transitions

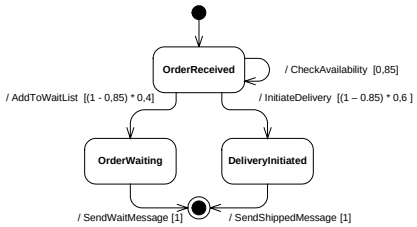
Probabilities for different control flows

- Add probabilities to transitions, if multiple choices



Example: Probabilities

- Estimate probabilities

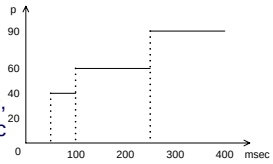


Time Consumption

- Need time consumption
  - Of internal computations
  - Of external calls
- No fix values
  - Influence of input parameters, variable values
  - Influence of platform, hardware, network, ...
  - No real time platforms

Distribution function

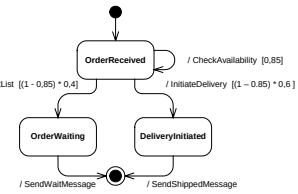
- Model time consumption of each node and transition with distribution functions
- Example:
  - 40% less than 50 msec,
  - 60% less than 100 msec,
  - 90% less than 250 msec



- Note: No upper bound given here, but possible

External Time Consumption

- Look up time consumptions for external calls, i.e. transitions
  - Will be available from performance predictions or measurements of the called services



External Time Consumption (2)

- CheckAvailability
  - Percentile 40 < 50 msec
  - Percentile 60 < 80 msec
  - Percentile 100 < 100 msec
- AddToWaitList and InitiateDelivery
  - Percentile 40 < 50 msec
  - Percentile 60 < 100 msec
  - Percentile 100 < 250 msec

External Time Consumption (3)

- SendWaitMessage
  - Percentile 60 < 100 msec
  - Percentile 100 < 200 msec
- SendShippedMessage
  - Percentile 60 < 100 msec
  - Percentile 100 < 200 msec

Internal time Consumption

- Estimate time consumption of internal computations
- Generally much lower than external
- Consider:
  - Workload, in abstract unit work unit
  - Device throughput
  - Both may vary (because of amount of data or device utilization)

Internal Time Consumption (2)

- Estimate operations of underlying code
- Assign into demand classes
  - High (e.g. a loop over all inputs, sorting, hard drive access)
  - Medium
  - Low (almost no computation needed)
- Assign work units according to class

Internal Time Consumption (3)

- Demand in example system
  - OrderReceived: Medium
  - DeliveryInitiated: Low
  - OrderWaiting: Low

Internal Time Consumption (4)

- Demand of classes:
  - Low: 24 work units
  - Medium: 48 work units
- Throughput:
  - 2 work units per msec

Internal Time Consumption (5)

- Resulting time consumptions:
- OrderWaiting
  - Percentile 100 < 12 msec
- DeliveryInitiated
  - Percentlie 100 < 12 msec
- OrderReceived
  - Percentile 100 < 24 msec

Using the tool

- Input data
  - Service Effect Automaton
  - Probabilities
  - Time Consumption
- Output
  - Distribution function for the analyzed service

Input File Format

- Regions:
  - states, startstate, finalstates, inputs, transitions, probabilities, time_consumption
  - All ended by a semicolon
  - Order must be preserved
- All names are case sensitive
- All names must be distinct
- Use # for comments, until end of line

Input: States

```
states: OrderReceived, OrderWaiting, DeliveryInitiated,
finalstate;

# One start state. Start state must be in states
startstate: OrderReceived;

# One or more final states. Each state must be in states.
finalstates: finalstate;
```


Input: Transitions

```
# List of input symbols
inputs: CheckAvailability, AddToWaitList, InitiateDelivery,
SendWaitMessage,SendShippedMessage;

# Transitions are defined: input=(source,destination)
# Input must be in inputs. Source and destination must be
# in states.
# Input is also the name of the transition.
transitions:
     CheckAvailability=(OrderReceived,OrderReceived),
     AddToWaitList=(OrderReceived,OrderWaiting),
     InitiateDelivery=(OrderReceived,DeliveryInitiated),
     SendWaitMessage=(OrderWaiting,finalstate),
     SendShippedMessage=(DeliveryInitiated,finalstate);
```


Input: Probabilities

```
# Probabilities for transitions.
# transition=probability
probabilities:
    CheckAvailability = 0.85,
    AddToWaitList=0.06,
    InitiateDelivery=0.09;
```

- Sum of probabilities for transitions leaving one state must be 1
- Not listed transitions have a probability of 1

Input: Distributions

```
# Time consumtion for a state or transition as QML aspect.
# stateOrTransition((percentiles),(times))
time_consumption:
    OrderReceived((100),(24)),
    OrderWaiting((100),(12)),
    DeliveryInitiated((100),(12)),
    CheckAvailability((40,60,100),(50,80,100)),
    AddToWaitList((40,60,100),(50,100,250)),
    InitiateDelivery((40,60,100),(50,100,250)),
    SendWaitMessage((60,100),(100,200)),
    SendShippedMessage((60,100),(100,200));
```


Input: Tips

- No commas in floating point numbers
- Use only integers for time consumptions.
- Alternatively, you may use the xml version of the tool.
  - Parses an xml file defining the SEA
  - See example in xml tool directory

Execute the tool

- You need the .NET framework 1.1
- Open a console
- Change into directory Palladio Performance Prediction
- Invoke tool with SEA and optionally the number of result values as parameter:

```
Palladio.Performance.Main.exe SampleSEA.dat {30}
```

- For linux, you can use the Mono framework
  - Runs .NET compatible programs

Output

- Output is written into Berechnet.txt

Berechnet.txt

- Some statistical reporting

```
Sum of all probabilities: 0,999999999999999
Minimum value: 51
Maximum value: 2545
Median: 567
UpperQuartil: 911
SamplingRate: 86
Expectancy: 600,474353049564
Variance: 170974,110874331
Deviation: 413,490158134787
```

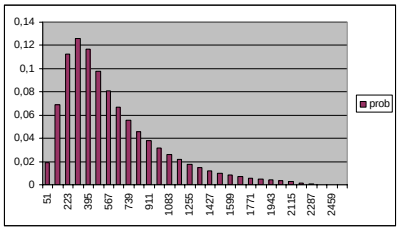

Berechnet.txt

- Probabilities for single intervals

```
Execution Time:      Probabilities:
51      0,0192418029273866
137     0,0689880476604597
223     0,112422482011755
309     0,125623647052229
395     0,116532685592899
481     0,0974960053488438
567     0,0808065427350571
653     0,066915032329015
739     0,055404292941819
:
:
```

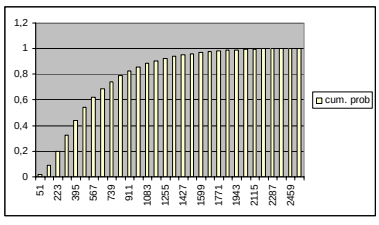

Probability Mass Function

- Probabilities for single intervals



Distribution Function

- Cumulative probabilities, transformed from PMF

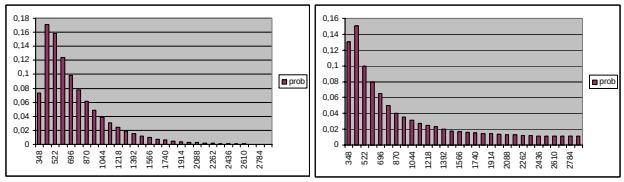


Results

- Distribution function of service in relation to called services performance
- Compositional approach: Result can be used as an input for analyzing services that call this service.
- Thus, system response time can be computed as a distribution function.

Evaluate the Results

- Which service has the better response time?



Evaluation

- Low mean response time may come with a high maximum response time.
- A lower maximum response time might have a much higher mean.
- Other values?

Evaluation

- Depends on performance requirements
- Different values to look at
  - Mean time
  - Maximum response time
  - Upper quartile
    - What response time with a probability of 0.75?
  - Variance / deviation
- Often outliers of interest

Evaluation

- Values for the Example:
- Upper quartiles: 911 msec
- Expectancy: 600,47 msec
- Median: 567 msec
- Approx. maximum response time: 2545 msec
- Variance: 170974,11 msec²
- Deviation: 413,49 msec

References

- [1] V. Firus, S. Becker und J. Happe, *Parametric Performance Contracts for QML-specified Software Components*, 2005, to be published in: Electronic Notes in Theoretical Computer Science.

A.4 Palladio Preparatory Exercise

Übungsblatt 2

Evaluation of Model-Driven Performance Prediction Techniques: Palladio Technique

Please work on the exercises alone. Send your solutions to Anne until June, 14th 2005.

Exercise 1:

- Install the .NET framework 1.1
- Unzip the Palladio tool (can be downloaded from Stud.IP)
- Make yourself familiar with it by analyzing the example.

Exercise 2:

The following sequence diagram (see appendix) models a system that finds flights for a user. It is similar to the system you know from exercise 1, but there are some differences.

The user specifies the time (including the date) and the start and destination airports. The FlightFinder component first requests all available AirlineWebServices from a service broker. Afterwards, it requests general information needed for communication from the web services, if they are not already known.
After obtaining the general information, the available flights are requested from each AirlineWebservice. This happens successively. After having requested the available flights, the flights are ordered by price. If there are at least three flights available from one of the requested airlines, the details for the three cheapest flight are requested from the corresponding airline (this a the difference to the system modelled in exercise 1).

All components are located on different servers, that are connected by the internet. The speed of the internet connection is included in the time consumption of the external services.

Additional control flow information

- The chance that an AirlineWebService is not known to the FlightFinder is 10 percent.
- The chance that no three flights are available is quite low, being 2 percent.
- The probability to exit the first loop that requests flights from the AirlineWebServices is 0.2

Time Consumption of External Services:

Called Service	Time Consumption
GetAirlineWebServices	Percentile 40 < 0.5 sec Percentile 75 < 1 sec Percentile 100 < 2 sec
GetAirlineInformation	Percentile 50 < 0.4 sec Percentile 75 < 1 sec Percentile 100 < 2 sec
GetFlights	Percentile 50 < 1.5 sec Percentile 75 < 2 sec Percentile 100 < 4 sec
GetDetails	Percentile 70 < 0.5 sec Percentile 100 < 1.5 sec

Remember that only integers can be inserted into the input SEA, so use msec for computations.

Time Consumption of Internal Services:
The processing demand of internal services ranges from 10 to 50 work units, 10 work units being a low demand, about 25 work units being a medium demand and 50 work units being a high demand. Assign the needed internal computation into these classes. The throughput of the server the service is deployed on is 100 work unit per second.

Your tasks:
The FindFlight Service is supposed to have a response time of less than 6 seconds in 75 percent of all uses. Will this performance goal be reached? What is the expected response time (i.e. mean response time) of the service?

To answer the questions, analyze the FlightFinder component's performance using the Palladio technique. Only the FindFlight service (below) is looked at. For analyzing the FlightFinder, use the given distribution functions of the called services and estimate the FlightFinder's internal computations. Equip a service effect automaton describing the behaviour of the FlightFinder with these distribution functions. Finally, use the Palladio tool to get a distribution function that describes the FlightFinder's performance. If you like, insert your solutions into a spread sheet tool like Open Office Calc and create a graph visualizing the distribution function.

Notes:

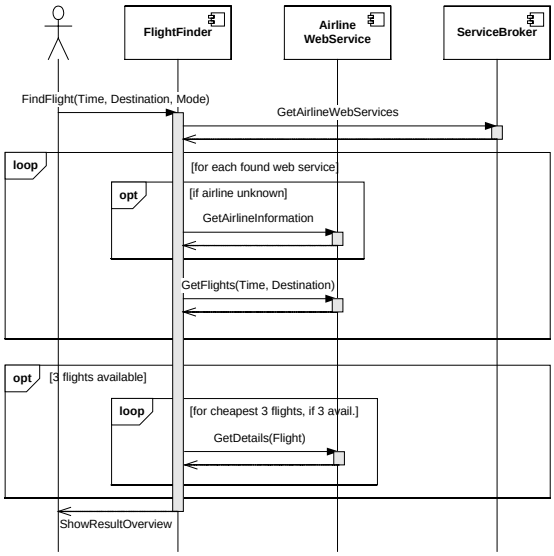
- You may change the number of values in the distribution functions so that they can be computed faster, but are less precise.
- Remember that all names of states and transitions must be distinct.
- Service Effect Automatons may have multiple final states If a state is a final state, the sum of the probabilities of the leaving transitions may be lower than one. Thus, the remaining probability is for ending in this state.
- The second loop is always iterated thrice or not at all. Thus, you cannot model the loop with a probability in the Service Effect Automaton, as a loop construct in the SEA does not have a fixed number of iterations.
- To model the demand of sorting, you may divide it and consider it demand for the requesting of the flights. This matches the fact that the time for sorting increases with more flights.
- Do not expect that the results are similar to the results of exercise 1, as different time consumptions are used.

Exercise 3:
Please answer these questions. They relate to both performance prediction technique.

1) Did you experience problems with a) the CB-SPE tool and b) the Palladio tool? If yes, please shortly describe the problems.
2) Assuming the tools worked fine, did you experience problems with a) the CB-SPE technique or b) the Palladio technique? If yes, please describe the problems.
3) Which of the techniques do you think better applicable? Why?
4) Do you have other comments on the exercises? If yes, please describe them.

Send your solutions, i.e. a graphical representation of the service effect automaton (in jpg, gif or another widely-used file format), the Palladio Tool input and output, the explanation of your decision and your ansers to the question in exercise 3 to anne.martens@informatik.uni-oldenburg.de. You do not need to include the distribution functions in the service effect automaton. Please include "Palladio exercise" and your name in the subject.

APPENDIX:



B Experimental Exercises

B.1 CB-SPE Exercise

Case Study: Evaluation of Model-Driven Performance Prediction Techniques

CB-SPE Technique

Your Name: _____

Exercise:

A new component-based webserver is designed. While designing, the developers discuss an architectural alternative that is supposed to improve the webservers performance. As you are familiar with performance engineering, you are asked to evaluate the alternatives.

The idea is to add a component that compresses the webserver's answers to requests. The resulting compressed data is readable by all current browsers. When used with plain HTML files and slow connections, this generally improves performance, because plain HTML can be compressed up to 80 percent. Additionally, the size of the messages is more important for the performance if slow connections are used. However, it is expected that this webserver will host many packed images, too, and that users have an adequate internet connection. As the compression of the files puts an additional processing load on the webserver, the designers want you to predict the performance of both alternatives and find out whether the compression really improves performance.

The developers provide you information on both design options, i.e. the unmodified webserver and the alternative webserver using compression. Only the `HTTPRequestProcessorTools` component is affected by the change. This component offers various services to the other components of the webserver. They all are deployed on a single server. For compression, only the service `SendContentDataToClient` is of importance. `SendContentDataToClient` receives a byte stream as a parameter, creates a header and sends both header and content byte stream to the user.

The performance of the other components does not change if compression is used and therefore is unimportant for the design decision. Thus, their actions are only roughly modelled.

If no compression is used, the `HTTPRequestProcessorTools` component consists of only one subcomponent, the `DefaultHTTPRequestProcessorTools` component. See component diagram 1 for the structure of this component. If compression is used, a second component `ZipHTTPRequestProcessorTools` is added to `HTTPRequestProcessorTools` using the pipe-and-filter pattern. The `ZipHTTPRequestProcessorTools` first processes a `SendContentDataToClient` call, and then forwards the compressed data to the `DefaultHTTPRequestProcessorTools` component, which sends it to the client. See component diagram 2 for the structure of the `HTTPRequestProcessorTools` with compression and sequence diagram 2 for the behavior of the `SendContentDataToClient` service.

The `ZipHTTPRequestProcessorTools` calls the `SharpZipLib` library to compress the data. Note that a byte stream is passed to the `SharpZipLib`, which cannot determine whether it contains already compressed data. Thus, the compression has a processing load that only depends on the size of the byte stream, not on the contained data.

The time the client needs to decompress the data is neglected in this analysis.

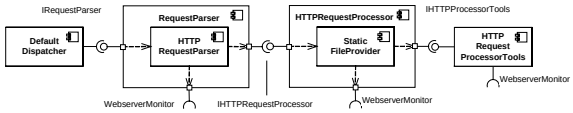
Additional information on performance data and timing behavior of called services is given in the tables in the appendix.

Your tasks:

- Estimate the demand and time consumption for calls and computations and list them in the tables in the appendix.
- For both design options, annotate the template sequence diagram with the RT-UML values. You find them on your PC at C:\CB_SPE\. See the notes to the templates below, too.
- Analyze the annotated diagram for one user. To do so, start the CB_SPE.EXE.bat in the C:\CB_SPE directory. Then, run the RAQS tool located at C:\CB_SPE\raqs\ and interpret the results. The RAQS input file is input.txt in the Output directory.
- What design option should be chosen? Why?
- Save your results, i.e. the zargo project and the results of the RAQS tool in the CB-SPE directory. All files should contain your name in their file names.

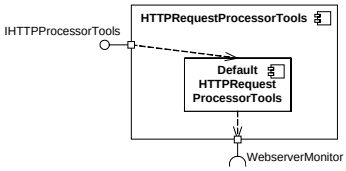
APPENDIX

The webserver's architecture (simplified):



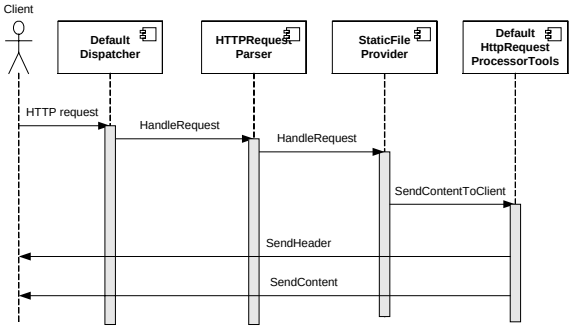
Alternative 1: No compression

Component Diagram 1:
`SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.



Sequence Diagram 1:
This diagram describes the behavior of the `DefaultHttpRequestProcessorTools` component without compression.

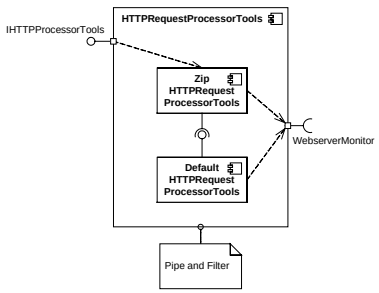
1) The `DefaultDispatcher` component listens to the ports specified in a configuration file. If a request is received, the `DefaultDispatcher` will create a new thread for handling this request. Within this thread, a new `HTTPRequestParser` is created and its `HandleRequest(Request)` method is called.
2) The `HTTPRequestParser` parses the HTTP information of the request and then calls the `HandleRequest(HTTPRequest)` method of a `HTTPRequestProcessor`, in our case the `StaticFileProvider` component.
3) The `StaticFileProvider` component retrieves the requested file from the file cache (no disk access needed) and converts it to a byte array. Finally, it calls `DefaultHTTPRequestProcessorTools'` `SendContentToClient` method, passing the byte array as a parameter.
4) The `DefaultHTTPRequestProcessorTools` component first converts the header to a byte array, too, and sends it to the client. Then, the byte array containing the requested file is sent.



Alternative 2: Compression

Component Diagram 2:

The `HTTPRequestProcessorTools` components are organised in a pipe-and-filter pattern. `SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.

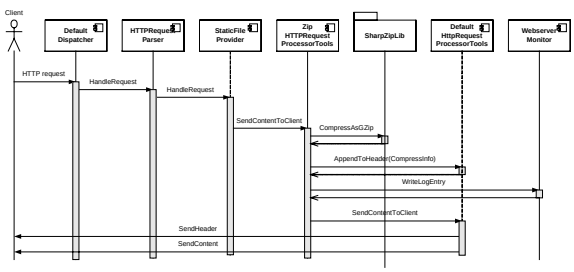


Sequence Diagram 2:

This diagram describes the behavior of the components if compression is used. The first components act like before, cf. sequence diagram 1, except that the `StaticFileProvider` component calls the `ZipHttpRequestProcessorTools` instead of the `DefaultHttpRequestProcessorTools`.

- The `ZipHttpRequestProcessorTools` component compresses the content data, appends the compression information to the header and writes a log entry. Afterwards, the `DefaultHttpRequestProcessorTools` component is called.
- Then the `DefaultHttpRequestProcessorTools` behaves like above.

Note that the content of the header is not compressed. Thus, the header is slightly bigger with the compression information.



Additional data:

You know that the demand of internal computation ranges between 0 and 7 work units. The throughput of the server is 1000 work units per second.

Note: You have to put a small load on the client machine, as otherwise RAQS will complain about a singular matrix, which cannot be analyzed. Set the client machines throughput to 500 and put a demand of 0,001 on first call of the user (HTTP request in zargo project).

Internal Computation	**Demand in work units (range 0-7)**
DefaultDispatcher (before calling HandleRequest)	
HTTPRequestParser (before calling HandleRequest)	
StaticFileProvider (before calling SendContentToClient)	
DefaultHTTPRequestProcessorTools (before calling SendHeader)	
DefaultHTTPRequestProcessorTools (before calling SendContent)	
ZipHTTPRequestProcessorTools (before compressing)	
ZipHTTPRequestProcessorTools (before appending the header)	
ZipHTTPRequestProcessorTools (before writing a log entry)	
ZipHTTPRequestProcessorTools (before calling SendContentToClient)	

The compression of a 15 KB file takes about 10 to 20 msec. Designers are not sure whether there is a linear relationship between file size and time to compress the file. Additionally, part of the compression call may be independent of the file size and thus take constant time. You have to estimate the demand for a 50KB file, which is the average file size. The demand for the other two external calls (`AppendToHeader, WriteLogEntry`) will probably take no longer than the internal computations. For `WriteLogEntry`, no disk access is needed.

External Call	**Demand in work units**
SharpZipLib.CompressAsGZip	
DefaultHTTPRequestProcessor.AppendToHeader	
WebserverMonitor.WriteLogEntry	

Network speed is 1Mbit per second. The average file size of the requested content is 50KB. The compression rate must be estimated. A standard header has a size of about 120 byte. A standard HTTP request has a size of about 130 byte. Both request and header are not compressed.

Compression	
Compression rate	
Size of requested file without compression	50 KB
Size of requested file after compression	KB
HTTP request size	130 byte
Header size without compression	120 byte
Header size with compression information	byte

Notes to the templates:

The `SendHeader` call from `DefaultRequestProcessorTools` is modeled with a return in the template sequence diagram. This is needed for CB-SPE, as the tool assumes parallel processing without this return call. Do not annotate this return call (you can just leave out the annotation field and go on with step 7).

If you experience problems, check the output.txt file in the Output directory of the CB-SPE tool. When running without problems, this file contains a list of all components, followed by the information of the deployment diagram and the sequence diagram, both with the RT-UML information. If the CB-SPE output does not help you, please ask.

Your interpretation of the results:

What design option should be chosen? Why?

B.2 Palladio Exercise

Case Study: Evaluation of Model-Driven Performance Prediction Techniques

Palladio Technique

Your Name: _____

Exercise:

A new component-based webserver is designed. While designing, the developers discuss an architectural alternative that is supposed to improve the webservers performance. As you are familiar with performance engineering, you are asked to evaluate the alternatives.

The idea is to add a component that compresses the webserver's answers to requests. The resulting compressed data is readable by all current browsers. When used with plain HTML files and slow connections, this generally improves performance, because plain HTML can be compressed up to 80 percent. Additionally, the size of the messages is more important for the performance if slow connections are used. However, it is expected that this webserver will host many packed images, too, and that users have an adequate internet connection. As the compression of the files puts an additional processing load on the webserver, the designers want you to predict the performance of both alternatives and find out whether the compression really improves performance.

The developers provide you information on both design options, i.e. the unmodified webserver and the alternative webserver using compression. Only the `HTTPRequestProcessorTools` component is affected by the change. This component offers various services to the other components of the webserver. They all are deployed on a single server. For compression, only the service `SendContentDataToClient` is of importance. `SendContentDataToClient` receives a byte stream as a parameter, creates a header and sends both header and content byte stream to the user.

The performance of the other components does not change if compression is used and therefore is unimportant for the design decision. Thus, their actions are only roughly modelled.

If no compression is used, the `HTTPRequestProcessorTools` component consists of only one subcomponent, the `DefaultHTTPRequestProcessorTools` component. See component diagram 1 for the structure of this component. If compression is used, a second component `ZipHTTPRequestProcessorTools` is added to `HTTPRequestProcessorTools` using the pipe-and-filter pattern. The `ZipHTTPRequestProcessorTools` first processes a `SendContentDataToClient` call, and then forwards the compressed data to the `DefaultHTTPRequestProcessorTools` component, which sends it to the client. See component diagram 2 for the structure of the `HTTPRequestProcessorTools` with compression and sequence diagram 2 for the behavior of the `SendContentDataToClient` service.

The `ZipHTTPRequestProcessorTools` calls the `SharpZipLib` library to compress the data. Note that a byte stream is passed to the `SharpZipLib`, which cannot determine whether it contains already compressed data. Thus, the compression has a processing load that only depends on the size of the byte stream, not on the contained data.

The time the client needs to decompress the data is neglected in this analysis.

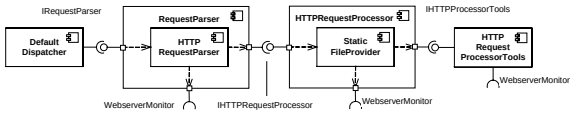
Additional information on performance data and timing behavior of called services is given in the tables in the appendix.

Your tasks:

- Create service effect automata describing the `SendContentToClient` service of both `DefaultHTTPRequestProcessorTools` and `ZipHTTPRequestProcessorTools`. Treat the sending of the header and the content to the client as an external call from the `DefaultHTTPRequestProcessorTools` component. Create graphical representations of these service effect automata either on this sheet or on your PC (write filename on this sheet). The service effect automata for the `DefaultDispatcher`, the `HTTPRequestParser` and the `StaticFileProvider` can be combined in one automaton (see below).
- Estimate the demand and time consumption for the nodes and edges and list them in the tables in the appendix.
- Create input files for the Palladio tool. Note that you need the results of an analysis of a service A before you can create the input for a service B that calls service A. For the different design options, you need different time consumptions for sending header and content in your input file for the `DefaultHTTPRequestProcessorTools` component, thus you have to create two input files for the `DefaultHTTPRequestProcessorTools` component. See notes to the Palladio tool below, too.
- Interpret the results of the Palladio tool. You find the Palladio tool and all needed files at C:\Palladio\.
- What design option should be chosen? Why? Consider the upper quartile value for your decision.
- Save your results, i.e. the three textual representations of the service effect automata (and the two graphical representations, if not on this sheet) as well as the results of the RAQS tool in the Palladio directory. All files should contain your name in their file names.

APPENDIX

The webserver's architecture (simplified):



Notes to the Palladio Tool:

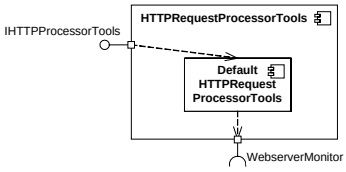
A Palladio input of ((100),(50)) means that all calls will be completed in 50 time units. It does not state, however, what the lowest possible time is. If you estimate a time consumption of 50 for a transition A, you want to model that calls need about 50 time units. By stating ((100),(50)) as a time consumption, the tool will assume a distribution whose maximum value is 50, but whose expected value will be significantly lower. Thus, you have to tighten the bounds for time consumption yourself: In this case, state ((1,100),(49,50)). Thus, only one percent of the calls will be completed in less than 49 time units. If your tool takes too long to analyze such an service effect automaton, try ((1,100),(45,50)), so that the tool can use greater sampling rates.

If you have analyzed one service and want to use the results as an external call's time consumption for another service, you can use the median x, upper quartile y and maximum value z in the output file and convert them into a distribution ((50,75,100),(x,y,z)). You may also add a lower bound like described above, to do so you have to look at the probability function the Palladio tool computes. Look up in which time 1 percent of the calls are completed by summing up the probabilties (Note that the Palladio tool computes a probability function, not a distribution function, thus you have to cumulate the probabilities to get the distribution function).

Have some patience while the Palladio tool analyzes the service effect automaton.

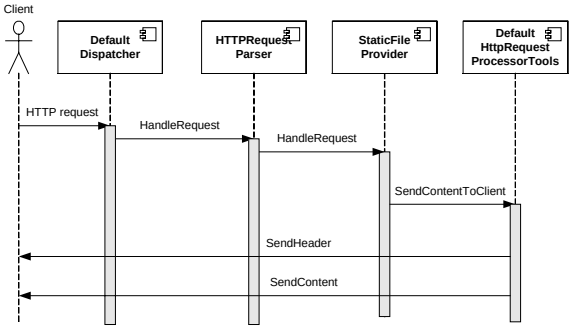
Alternative 1: No compression

Component Diagram 1:
`SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.



Sequence Diagram 1:
This diagram describes the behavior of the `DefaultHttpRequestProcessorTools` component without compression.

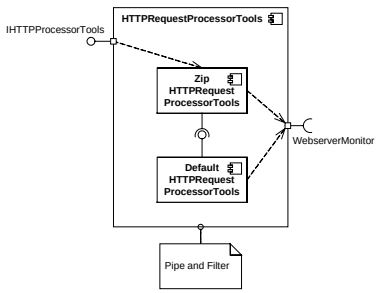
1) The `DefaultDispatcher` component listens to the ports specified in a configuration file. If a request is received, the `DefaultDispatcher` will create a new thread for handling this request. Within this thread, a new `HTTPRequestParser` is created and its `HandleRequest(Request)` method is called.
2) The `HTTPRequestParser` parses the HTTP information of the request and then calls the `HandleRequest(HTTPRequest)` method of a `HTTPRequestProcessor`, in our case the `StaticFileProvider` component.
3) The `StaticFileProvider` component retrieves the requested file from the file cache (no disk access needed) and converts it to a byte array. Finally, it calls `DefaultHTTPRequestProcessorTools'` `SendContentToClient` method, passing the byte array as a parameter.
4) The `DefaultHTTPRequestProcessorTools` component first converts the header to a byte array, too, and sends it to the client. Then, the byte array containing the requested file is sent.



Alternative 2: Compression

Component Diagram 2:

The `HTTPRequestProcessorTools` components are organised in a pipe-and-filter pattern. `SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.

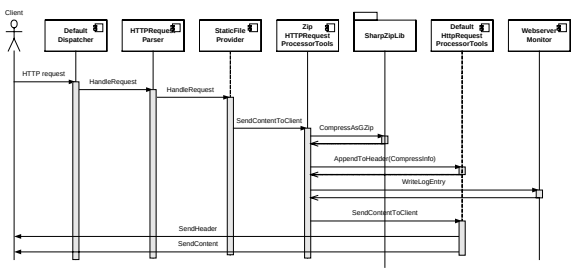


Sequence Diagram 2:

This diagram describes the behavior of the components if compression is used. The first components act like before, cf. sequence diagram 1, except that the `StaticFileProvider` component calls the `ZipHttpRequestProcessorTools` instead of the `DefaultHttpRequestProcessorTools`.

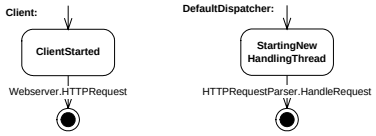
- The `ZipHttpRequestProcessorTools` component compresses the content data, appends the compression information to the header and writes a log entry. Afterwards, the `DefaultHttpRequestProcessorTools` component is called.
- Then the `DefaultHttpRequestProcessorTools` behaves like above.

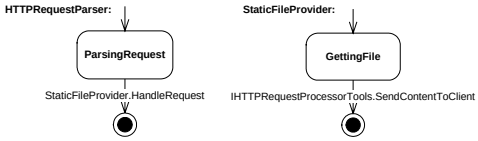
Note that the content of the header is not compressed. Thus, the header is slightly bigger with the compression information.



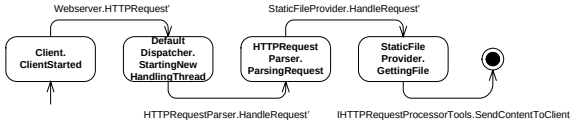
Service Effect Automata for Client, DefaultDispatcher, HTTPRequestParser and StaticFileProvider:

The service effect automata for the `Client, DefaultDispatcher, HTTPRequestParser` and `StaticFileProvider` are simple, as they contain just one state and one call.





To make analysis faster, these service effect automata can be combined into one:



Here, the calls marked with an apostrophe model only the begin of an external call. Thus, `HTTPRequestParser.HandleRequest'` and `StaticFileProvider.HandleRequest'` are epsilon transitions with time consumption 0. For `Webserver.HTTPRequest'`, however, you need to consider the time for sending the request to the webserver over the network. You find this service effect automaton in the Palladio directory, you only have to add the time consumptions.

The time consumption for the `IHTTPRequestProcessorTools.HandleRequest` call (either to `DefaultHTTPRequestProcessor` or `ZipHTTPRequestProcessor`, depending on design decision) has to be inserted after having analyzed the service effect automaton for the called service.
The sending of the header and content to the client should be modeled as an external call in the `DefaultHTTPRequestProcessorTools` component.

Additional data:

You know that the demand of internal computation ranges between 0 and 7 work units. The throughput of the server is 1000 work units per second.

Internal Computation	Demand in work units (range 0-7)
DefaultDispatcher (before calling HandleRequest)	
HTTPRequestParser (before calling HandleRequest)	
StaticFileProvider (before calling SendContentToClient)	
DefaultHTTPRequestProcessor (before calling SendHeader)	
DefaultHTTPRequestProcessor (before calling SendContent)	
ZipHTTPRequestProcessor (before compressing)	
ZipHTTPRequestProcessor (before appending the header)	
ZipHTTPRequestProcessor (before writing a log entry)	
ZipHTTPRequestProcessor (before calling SendContentToClient)	

The compression of a 15 KB file takes about 10 to 20 msec. Designers are not sure whether there is a linear relationship between file size and time to compress the file. Additionally, part of the compression call may be independent of the file size and thus take constant time. You have to estimate the time for a 50KB file, which is the average file size. The time consumption for the other two external calls (`AppendToHeader, WriteLogEntry`) will probably take no longer than the internal computations. For WriteLogEntry, no disk access is needed.

External Call	Time Consumption in ms
SharpLibZip.CompressAsGZip	
DefaultHTTPRequestProcessor.AppendToHeader	
WebserverMonitor.WriteLogEntry	

Network speed is 1Mbit per second. The average file size of the requested content is 50KB. The compression rate must be estimated. A standard header has a size of about 120 byte. A standard HTTP request has a size of about 130 byte. Both request and header are not compressed.

Compression	
Compression rate	
File size of requested content after compression	KB
Time to transfer requested file with compression	ms
Time to transfer requested file without compression	ms
Time to transfer header with compression information	ms
Time to transfer header without compression information	ms
Time to transfer HTTP request	ms

Service effect automata

DefaultHTTPRequestProcessorTools.SendContentToClient

ZipHTTPRequestProcessorTools.SendContentToClient

Your interpretation of the results:

What design option should be chosen? Why?

B.3 Exercise for Comparison Group

Case Study: Evaluation of Model-Driven Performance Prediction Techniques

No prediction technique

Your Name: _____

Exercise:

A new component-based webserver is designed. While designing, the developers discuss an architectural alternative that is supposed to improve the webservers performance. As you are familiar with performance engineering, you are asked to evaluate the alternatives.

The idea is to add a component that compresses the webserver's answers to requests. The resulting compressed data is readable by all current browsers. When used with plain HTML files and slow connections, this generally improves performance, because plain HTML can be compressed up to 80 percent. Additionally, the size of the messages is more important for the performance if slow connections are used. However, it is expected that this webserver will host many packed images, too, and that users have an adequate internet connection. As the compression of the files puts an additional processing load on the webserver, the designers want you to predict the performance of both alternatives and find out whether the compression really improves performance.

The developers provide you information on both design options, i.e. the unmodified webserver and the alternative webserver using compression. Only the `HTTPRequestProcessorTools` component is affected by the change. This component offers various services to the other components of the webserver. They all are deployed on a single server. For compression, only the service `SendContentDataToClient` is of importance. `SendContentDataToClient` receives a byte stream as a parameter, creates a header and sends both header and content byte stream to the user.

The performance of the other components does not change if compression is used and therefore is unimportant for the design decision. Thus, their actions are only roughly modelled.

If no compression is used, the `HTTPRequestProcessorTools` component consists of only one subcomponent, the `DefaultHTTPRequestProcessorTools` component. See component diagram 1 for the structure of this component. If compression is used, a second component `ZipHTTPRequestProcessorTools` is added to `HTTPRequestProcessorTools` using the pipe-and-filter pattern. The `ZipHTTPRequestProcessorTools` first processes a `SendContentDataToClient` call, and then forwards the compressed data to the `DefaultHTTPRequestProcessorTools` component, which sends it to the client. See component diagram 2 for the structure of the `HTTPRequestProcessorTools` with compression and sequence diagram 2 for the behavior of the `SendContentDataToClient` service.

The `ZipHTTPRequestProcessorTools` calls the `SharpZipLib` library to compress the data. Note that a byte stream is passed to the `SharpZipLib`, which cannot determine whether it contains already compressed data. Thus, the compression has a processing load that only depends on the size of the byte stream, not on the contained data.

The time the client needs to decompress the data is neglected in this analysis.

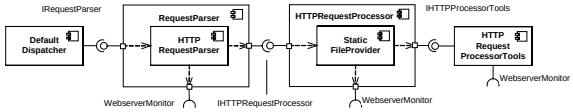
Additional information on performance data and timing behavior of called services is given in the tables in the appendix.

Your tasks:

- Please calculate or estimate the response times of both design options.
- What design option should be chosen? Why?

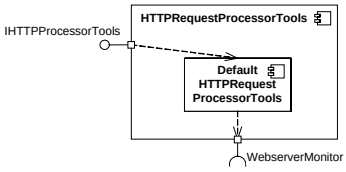
APPENDIX

The webserver's architecture (simplified):



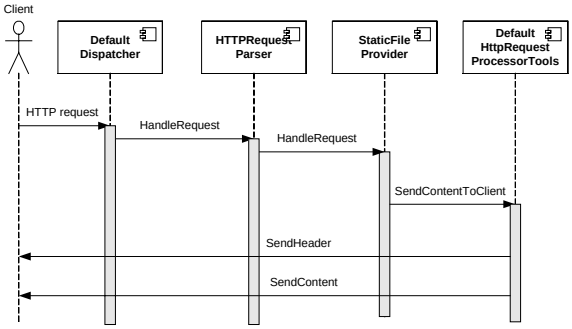
Alternative 1: No compression

Component Diagram 1:
`SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.



Sequence Diagram 1:
This diagram describes the behavior of the `DefaultHttpRequestProcessorTools` component without compression.

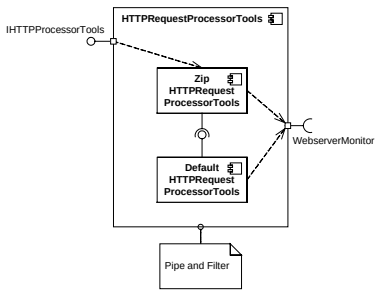
1) The `DefaultDispatcher` component listens to the ports specified in a configuration file. If a request is received, the `DefaultDispatcher` will create a new thread for handling this request. Within this thread, a new `HTTPRequestParser` is created and its `HandleRequest(Request)` method is called.
2) The `HTTPRequestParser` parses the HTTP information of the request and then calls the `HandleRequest(HTTPRequest)` method of a `HTTPRequestProcessor`, in our case the `StaticFileProvider` component.
3) The `StaticFileProvider` component retrieves the requested file from the file cache (no disk access needed) and converts it to a byte array. Finally, it calls `DefaultHTTPRequestProcessorTools'` `SendContentToClient` method, passing the byte array as a parameter.
4) The `DefaultHTTPRequestProcessorTools` component first converts the header to a byte array, too, and sends it to the client. Then, the byte array containing the requested file is sent.



Alternative 2: Compression

Component Diagram 2:

The `HTTPRequestProcessorTools` components are organised in a pipe-and-filter pattern. `SendContentDataToClient` is a service of the `IHTTPProcessorTools` interface.

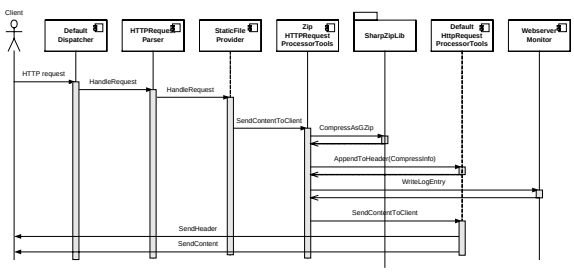


Sequence Diagram 2:

This diagram describes the behavior of the components if compression is used. The first components act like before, cf. sequence diagram 1, except that the `StaticFileProvider` component calls the `ZipHttpRequestProcessorTools` instead of the `DefaultHttpRequestProcessorTools`.

- The `ZipHttpRequestProcessorTools` component compresses the content data, appends the compression information to the header and writes a log entry. Afterwards, the `DefaultHttpRequestProcessorTools` component is called.
- Then the `DefaultHttpRequestProcessorTools` behaves like above.

Note that the content of the header is not compressed. Thus, the header is slightly bigger with the compression information.



Additional data:

You know that the demand of internal computation ranges between 0 and 7 work units. The throughput of the server is 1000 work units per second.

Internal Computation	**Demand in work units (range 0-7)**
DefaultDispatcher (before calling HandleRequest)	
HTTPRequestParser (before calling HandleRequest)	
StaticFileProvider (before calling SendContentToClient)	
DefaultHTTPRequestProcessor (before calling SendHeader)	
DefaultHTTPRequestProcessor (before calling SendContent)	
ZipHTTPRequestProcessor (before compressing)	
ZipHTTPRequestProcessor (before appending the header)	
ZipHTTPRequestProcessor (before writing a log entry)	
ZipHTTPRequestProcessor (before calling SendContentToClient)	

The compression of a 15 KB file takes about 10 to 20 msec. Designers are not sure whether there is a linear relationship between file size and time to compress the file. Additionally, part of the compression call may be independent of the file size and thus take constant time. You have to estimate the time for a 50KB file, which is the average file size. The time consumption for the other two external calls (`AppendToHeader, WriteLogEntry`) will probably take no longer than the internal computations. For WriteLogEntry, no disk access is needed.

External Call	**Time Consumption in ms**
SharpLibZip.CompressAsGZip	
DefaultHTTPRequestProcessor.AppendToHeader	
WebserverMonitor.WriteLogEntry	

Network speed is 1Mbit per second. The average file size of the requested content is 50KB. The compression rate must be estimated. A standard header has a size of about 120 byte. A standard HTTP request has a size of about 130 byte. Both request and header are not compressed.

Compression	
Compression rate	
File size of requested content after compression	KB
Time to transfer requested file with compression	ms
Time to transfer requested file without compression	ms
Time to transfer header with compression information	ms
Time to transfer header without compression information	ms
Time to transfer HTTP request	ms

Your solution:

Response Time	
Response time without compression	
Response time with compression	

Room for Notes

Your interpretation of the results:

What design option should be chosen? Why?

Time you needed to analyze the system: _____

C Resulting Data

C.1 Predictions of the Participants

Results of the performance predictions

CB-SPE results

	Response time for design option...		
Participant	without compression	with compression	compression rate
1	421	285	50
2	402,017	322	20
3	417	205	60
4	418	319	40
5	415	373	50
6	412	149	80
7	416	273	50
8	419	112	80
9	414	159	80
10	431	318	30

Palladio results

	Distribution for design option...				
	without compression		with compression		
Participant	Average	Upper Quartile	Average	Upper Quartile	Compression Rate
1	374,95	389,33	224,6425	285,25	40
2	398,9852	398,9852	375,12	375,12	20
3	398,045	401	118,75	162	80
4	312	415	263,5	350	30
5	346,25	480	375	500	10
6	418,925	418,925	295,545	295,545	50
7	285,8	404	240	338	40
8	313,9147627	430,858	196,5295784	271,039	50
9	403,975	407	264,375	334	35
10	419,68	621	255,86	362	50
11	318,25	422	261,5	346	40

the distribution functions are too large to be displayed here.

C.2 Results of Measurements

Measurements Web Server

uncompressed 50 KB

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.384	135	00:00:04	52
2	0.40300003	128	00:00:05	52
2	0	0	00:00:06	0
3	0.38300002	135	00:00:07	52
4	0.38300002	135	00:00:08	52
4	0	0	00:00:09	0
5	0.384	135	00:00:10	52
6	0.386	134	00:00:11	52
7	0.41300002	125	00:00:12	52
7	0	0	00:00:13	0
8	0.402	129	00:00:14	52
9	0.384	135	00:00:15	52
9	0	0	00:00:16	0
10	0.38300002	135	00:00:17	52
11	0.38300002	135	00:00:18	52
12	0.393	132	00:00:19	52
12	0	0	00:00:20	0
13	0.384	135	00:00:21	52
14	0.38300002	135	00:00:22	52
14	0	0	00:00:23	0
15	0.38300002	135	00:00:24	52
16	0.38700002	134	00:00:25	52
17	0.41400003	125	00:00:26	52
17	0	0	00:00:27	0
18	0.393	132	00:00:28	52
19	0.38300002	135	00:00:29	52
19	0	0	00:00:30	0
20	0.38300002	135	00:00:31	52
21	0.384	135	00:00:32	52
22	0.386	134	00:00:33	52
22	0	0	00:00:34	0
23	0.38300002	135	00:00:35	52
24	0.38300002	135	00:00:36	52
24	0	0	00:00:37	0
25	0.384	135	00:00:38	52
26	0.386	134	00:00:39	52
27	0.41300002	125	00:00:40	52
27	0	0	00:00:41	0
28	0.384	135	00:00:42	52
29	0.384	135	00:00:43	52
29	0	0	00:00:44	0
30	0.38300002	135	00:00:45	52
31	0.397	130	00:00:46	52
32	0.423	122	00:00:47	52
32	0	0	00:00:48	0
33	0.384	135	00:00:49	52
34	0.38300002	135	00:00:50	52
34	0	0	00:00:51	0
35	0.38300002	135	00:00:52	52
36	0.38300002	135	00:00:53	52
37	0.40300003	128	00:00:54	52
37	0	0	00:00:55	0
38	0.38300002	135	00:00:56	52
39	0.38300002	135	00:00:57	52
39	0	0	00:00:58	0
40	0.384	135	00:00:59	52
41	0.38700002	134	00:01:00	52
42	0.41300002	125	00:01:01	52
42	0	0	00:01:02	0
43	0.38300002	135	00:01:03	52
44	0.384	135	00:01:04	52
44	0	0	00:01:05	0
45	0.38300002	135	00:01:06	52
46	0.384	135	00:01:07	52
47	0.41300002	125	00:01:08	52
47	0	0	00:01:09	0
48	0.38500002	134	00:01:10	52
49	0.38300002	135	00:01:11	52
49	0	0	00:01:12	0
50	0.38300002	135	00:01:13	52
51	0.38500002	134	00:01:14	52
52	0.41400003	125	00:01:15	52
52	0	0	00:01:16	0
53	0.38300002	135	00:01:17	52
54	0.38300002	135	00:01:18	52
54	0	0	00:01:19	0
55	0.384	135	00:01:20	52
56	0.38700002	134	00:01:21	52
57	0.41300002	125	00:01:22	52
57	0	0	00:01:23	0
58	0.402	129	00:01:24	52
59	0.384	135	00:01:25	52
59	0	0	00:01:26	0
60	0.38300002	135	00:01:27	52
61	0.38300002	135	00:01:28	52
62	0.393	132	00:01:29	52
62	0	0	00:01:30	0
63	0.384	135	00:01:31	52
64	0.38300002	135	00:01:32	52
64	0	0	00:01:33	0
65	0.38300002	135	00:01:34	52
66	0.386	134	00:01:35	52
67	0.41400003	125	00:01:36	52
67	0	0	00:01:37	0
68	0.38300002	135	00:01:38	52
69	0.38300002	135	00:01:39	52
69	0	0	00:01:40	0
70	0.38300002	135	00:01:41	52
71	0.384	135	00:01:42	52
71	0	0	00:01:43	0
72	0.386	134	00:01:44	52
73	0.38300002	135	00:01:45	52
74	0.38300002	135	00:01:46	52
74	0	0	00:01:47	0
75	0.384	135	00:01:48	52
76	0.384	135	00:01:49	52
76	0	0	00:01:50	0
77	0.41300002	125	00:01:51	52
78	0.384	135	00:01:52	52
79	0.38300002	135	00:01:53	52
79	0	0	00:01:54	0
80	0.38300002	135	00:01:55	52
81	0.38300002	135	00:01:56	52
81	0	0	00:01:57	0
82	0.38700002	134	00:01:58	52
83	0.38300002	135	00:01:59	52
84	0.38300002	135	00:02:00	52
84	0	0	00:02:01	0
85	0.38300002	135	00:02:02	52
86	0.384	135	00:02:03	52
86	0	0	00:02:04	0
87	0.41300002	125	00:02:05	52
88	0.38300002	135	00:02:06	52
89	0.38300002	135	00:02:07	52
89	0	0	00:02:08	0
90	0.384	135	00:02:09	52
91	0.388	133	00:02:10	52
91	0	0	00:02:11	0
92	0.38700002	134	00:02:12	52
93	0.384	135	00:02:13	52
94	0.384	135	00:02:14	52
94	0	0	00:02:15	0
95	0.38300002	135	00:02:16	52
96	0.38300002	135	00:02:17	52
96	0	0	00:02:18	0
97	0.41300002	125	00:02:19	52
98	0.384	135	00:02:20	52
99	0.38300002	135	00:02:21	52
99	0	0	00:02:22	0
100	0.38300002	135	00:02:23	52
101	0.384	135	00:02:24	52
101	0	0	00:02:25	0
102	0.41400003	125	00:02:26	52
103	0.38500002	134	00:02:27	52
104	0.38300002	135	00:02:28	52
104	0	0	00:02:29	0
105	0.384	135	00:02:30	52
106	0.38500002	134	00:02:31	52
106	0	0	00:02:32	0
107	0.41300002	125	00:02:33	52
108	0.384	135	00:02:34	52
109	0.384	135	00:02:35	52
109	0	0	00:02:36	0
110	0.38300002	135	00:02:37	52
111	0.386	134	00:02:38	52
111	0	0	00:02:39	0
112	0.41300002	125	00:02:40	52
113	0.384	135	00:02:41	52
114	0.38300002	135	00:02:42	52
114	0	0	00:02:43	0
115	0.38300002	135	00:02:44	52
116	0.386	134	00:02:45	52
116	0	0	00:02:46	0
117	0.41400003	125	00:02:47	52
118	0.38300002	135	00:02:48	52
119	0.38300002	135	00:02:49	52
119	0	0	00:02:50	0
120	0.384	135	00:02:51	52
121	0.386	134	00:02:52	52
121	0	0	00:02:53	0
122	0.41300002	125	00:02:54	52
123	0.38300002	135	00:02:55	52
124	0.384	135	00:02:56	52
124	0	0	00:02:57	0
125	0.38300002	135	00:02:58	52
126	0.384	135	00:02:59	52
126	0	0	00:03:00	0
127	0.41300002	125	00:03:01	52
128	0.4	129	00:03:02	52
129	0.38300002	135	00:03:03	52
129	0	0	00:03:04	0
130	0.384	135	00:03:05	52
131	0.38300002	135	00:03:06	52
131	0	0	00:03:07	0
132	0.41300002	125	00:03:08	52
133	0.38500002	134	00:03:09	52
134	0.38300002	135	00:03:10	52
134	0	0	00:03:11	0
135	0.38300002	135	00:03:12	52
136	0.38300002	135	00:03:13	52
136	0	0	00:03:14	0
137	0.41400003	125	00:03:15	52
138	0.38300002	135	00:03:16	52
139	0.38300002	135	00:03:17	52
139	0	0	00:03:18	0
140	0.38300002	135	00:03:19	52
141	0.384	135	00:03:20	52
141	0	0	00:03:21	0
142	0.41300002	125	00:03:22	52
143	0.38300002	135	00:03:23	52
144	0.38300002	135	00:03:24	52
144	0	0	00:03:25	0
145	0.384	135	00:03:26	52
146	0.386	134	00:03:27	52
146	0	0	00:03:28	0
147	0.386	134	00:03:29	52
148	0.38300002	135	00:03:30	52
149	0.384	135	00:03:31	52
149	0	0	00:03:32	0
150	0.38300002	135	00:03:33	52
151	0.386	134	00:03:34	52
151	0	0	00:03:35	0
152	0.41300002	125	00:03:36	52
153	0.384	135	00:03:37	52
154	0.38300002	135	00:03:38	52
154	0	0	00:03:39	0
155	0.38300002	135	00:03:40	52
156	0.38700002	134	00:03:41	52
156	0	0	00:03:42	0
157	0.41400003	125	00:03:43	52
158	0.38200003	135	00:03:44	52
159	0.38300002	135	00:03:45	52
159	0	0	00:03:46	0
160	0.384	135	00:03:47	52
161	0.384	135	00:03:48	52
161	0	0	00:03:49	0
162	0.41300002	125	00:03:50	52
163	0.38300002	135	00:03:51	52
164	0.384	135	00:03:52	52
164	0	0	00:03:53	0
165	0.38300002	135	00:03:54	52
166	0.386	134	00:03:55	52
166	0	0	00:03:56	0
167	0.41300002	125	00:03:57	52
168	0.384	135	00:03:58	52
169	0.38300002	135	00:03:59	52
169	0	0	00:04:00	0
170	0.38300002	135	00:04:01	52
171	0.38500002	134	00:04:02	52
171	0	0	00:04:03	0
172	0.41400003	125	00:04:04	52
173	0.38300002	135	00:04:05	52
173	0	0	00:04:06	0
174	0.38300002	135	00:04:07	52
175	0.384	135	00:04:08	52
176	0.38500002	134	00:04:09	52
176	0	0	00:04:10	0
177	0.41300002	125	00:04:11	52
178	0.38300002	135	00:04:12	52
178	0	0	00:04:13	0
179	0.384	135	00:04:14	52
180	0.38300002	135	00:04:15	52
181	0.38500002	134	00:04:16	52
181	0	0	00:04:17	0
182	0.41300002	125	00:04:18	52
183	0.384	135	00:04:19	52
183	0	0	00:04:20	0
184	0.38300002	135	00:04:21	52
185	0.38300002	135	00:04:22	52
186	0.386	134	00:04:23	52
186	0	0	00:04:24	0
187	0.41300002	125	00:04:25	52
188	0.38300002	135	00:04:26	52
188	0	0	00:04:27	0
189	0.38300002	135	00:04:28	52
190	0.38300002	135	00:04:29	52
191	0.39400002	131	00:04:30	52
191	0	0	00:04:31	0
192	0.40300003	128	00:04:32	52
193	0.38300002	135	00:04:33	52
193	0	0	00:04:34	0
194	0.38300002	135	00:04:35	52
195	0.384	135	00:04:36	52
196	0.38500002	134	00:04:37	52
196	0	0	00:04:38	0
197	0.41300002	125	00:04:39	52
198	0.384	135	00:04:40	52
198	0	0	00:04:41	0
199	0.384	135	00:04:42	52
200	0.38300002	135	00:04:43	52
201	0.38500002	134	00:04:44	52
201	0	0	00:04:45	0
202	0.41300002	125	00:04:46	52
203	0.38300002	135	00:04:47	52
203	0	0	00:04:48	0
204	0.38300002	135	00:04:49	52
205	0.38300002	135	00:04:50	52
206	0.386	134	00:04:51	52
206	0	0	00:04:52	0
207	0.41300002	125	00:04:53	52
208	0.38300002	135	00:04:54	52
208	0	0	00:04:55	0
209	0.38300002	135	00:04:56	52
210	0.384	135	00:04:57	52
211	0.38700002	134	00:04:58	52
211	0	0	00:04:59	0
212	0.38700002	134	00:05:00	52
213	0.629	82	00:05:01	52
213	0	0	00:05:02	0
214	0.39800003	130	00:05:03	52
215	0.38300002	135	00:05:04	52
215	0	0	00:05:05	0
216	0.397	130	00:05:06	52
217	0.38300002	135	00:05:07	52
218	0.38300002	135	00:05:08	52
218	0	0	00:05:09	0
219	0.38300002	135	00:05:10	52
220	0.38300002	135	00:05:11	52
220	0	0	00:05:12	0
221	0.384	135	00:05:13	52
222	0.384	135	00:05:14	52
223	0.393	132	00:05:15	52
223	0	0	00:05:16	0
224	0.38300002	135	00:05:17	52
225	0.384	135	00:05:18	52
225	0	0	00:05:19	0
226	0.38300002	135	00:05:20	52
227	0.38300002	135	00:05:21	52
228	0.38700002	134	00:05:22	52
228	0	0	00:05:23	0
229	0.384	135	00:05:24	52
230	0.384	135	00:05:25	52
230	0	0	00:05:26	0
231	0.38300002	135	00:05:27	52
232	0.38300002	135	00:05:28	52
233	0.41300002	125	00:05:29	52
233	0	0	00:05:30	0
234	0.38300002	135	00:05:31	52
235	0.38300002	135	00:05:32	52
235	0	0	00:05:33	0
236	0.384	135	00:05:34	52
237	0.381	136	00:05:35	52
238	0.41300002	125	00:05:36	52
238	0	0	00:05:37	0
239	0.38300002	135	00:05:38	52
240	0.384	135	00:05:39	52
240	0	0	00:05:40	0
241	0.38300002	135	00:05:41	52
242	0.38300002	135	00:05:42	52
243	0.41300002	125	00:05:43	52
243	0	0	00:05:44	0
244	0.384	135	00:05:45	52
245	0.38300002	135	00:05:46	52
245	0	0	00:05:47	0
246	0.38300002	135	00:05:48	52
247	0.38500002	134	00:05:49	52
248	0.41400003	125	00:05:50	52
248	0	0	00:05:51	0
249	0.38300002	135	00:05:52	52
250	0.38300002	135	00:05:53	52
250	0	0	00:05:54	0
251	0.38300002	135	00:05:55	52
252	0.386	134	00:05:56	52
253	0.38700002	134	00:05:57	52
253	0	0	00:05:58	0
254	0.38300002	135	00:05:59	52
255	0.38300002	135	00:06:00	52
255	0	0	00:06:01	0
256	0.38300002	135	00:06:02	52
257	0.397	130	00:06:03	52
257	0	0	00:06:04	0
258	0.38300002	135	00:06:05	52
259	0.38300002	135	00:06:06	52
260	0.384	135	00:06:07	52
260	0	0	00:06:08	0
261	0.40300003	128	00:06:09	52
262	0.393	132	00:06:10	52
262	0	0	00:06:11	0
263	0.38300002	135	00:06:12	52
264	0.384	135	00:06:13	52
265	0.38300002	135	00:06:14	52
265	0	0	00:06:15	0
266	0.38300002	135	00:06:16	52
267	0.38300002	135	00:06:17	52
267	0	0	00:06:18	0
268	0.384	135	00:06:19	52
269	0.38300002	135	00:06:20	52
270	0.38300002	135	00:06:21	52
270	0	0	00:06:22	0
271	0.38300002	135	00:06:23	52
272	0.384	135	00:06:24	52
272	0	0	00:06:25	0
273	0.384	135	00:06:26	52
274	0.41300002	125	00:06:27	52
275	0.38300002	135	00:06:28	52
275	0	0	00:06:29	0
276	0.384	135	00:06:30	52
277	0.38300002	135	00:06:31	52
277	0	0	00:06:32	0
278	0.38300002	135	00:06:33	52
279	0.41400003	125	00:06:34	52
280	0.384	135	00:06:35	52
280	0	0	00:06:36	0
281	0.38300002	135	00:06:37	52
282	0.38300002	135	00:06:38	52
282	0	0	00:06:39	0
283	0.384	135	00:06:40	52
284	0.41400003	125	00:06:41	52
285	0.38300002	135	00:06:42	52
285	0	0	00:06:43	0
286	0.38300002	135	00:06:44	52
287	0.384	135	00:06:45	52
287	0	0	00:06:46	0
288	0.38700002	134	00:06:47	52
289	0.41700003	124	00:06:48	52
290	0.38300002	135	00:06:49	52
290	0	0	00:06:50	0
291	0.38200003	135	00:06:51	52
292	0.38300002	135	00:06:52	52
292	0	0	00:06:53	0
293	0.38300002	135	00:06:54	52
294	0.41300002	125	00:06:55	52
295	0.38500002	134	00:06:56	52
295	0	0	00:06:57	0
296	0.38300002	135	00:06:58	52
297	0.38300002	135	00:06:59	52
297	0	0	00:07:00	0
298	0.386	134	00:07:01	52
299	0.38700002	134	00:07:02	52
300	0.38300002	135	00:07:03	52
300	0	0	00:07:04	0
301	0.38300002	135	00:07:05	52
302	0.38300002	135	00:07:06	52
302	0	0	00:07:07	0
303	0.384	135	00:07:08	52
304	0.41300002	125	00:07:09	52
304	0	0	00:07:10	0
305	0.38300002	135	00:07:11	52
306	0.384	135	00:07:12	52
307	0.38300002	135	00:07:13	52
307	0	0	00:07:14	0
308	0.388	133	00:07:15	52
309	0.38700002	134	00:07:16	52
309	0	0	00:07:17	0
310	0.384	135	00:07:18	52
311	0.38300002	135	00:07:19	52
312	0.38300002	135	00:07:20	52
312	0	0	00:07:21	0

313	0.384	135	00:07:22	52
314	0.418	124	00:07:23	52
314	0	0	00:07:24	0
315	0.38300002	135	00:07:25	52
316	0.38300002	135	00:07:26	52
317	0.38300002	135	00:07:27	52
317	0	0	00:07:28	0
318	0.38900003	133	00:07:29	52
319	0.386	134	00:07:30	52
319	0	0	00:07:31	0
320	0.38300002	135	00:07:32	52
321	0.38300002	135	00:07:33	52
322	0.384	135	00:07:34	52
322	0	0	00:07:35	0
323	0.39800003	130	00:07:36	52
324	0.41300002	125	00:07:37	52
324	0	0	00:07:38	0
325	0.38300002	135	00:07:39	52
326	0.384	135	00:07:40	52
327	0.38300002	135	00:07:41	52
327	0	0	00:07:42	0
328	0.386	134	00:07:43	52
329	0.38700002	134	00:07:44	52
329	0	0	00:07:45	0
330	0.384	135	00:07:46	52
331	0.38300002	135	00:07:47	52
332	0.384	135	00:07:48	52
332	0	0	00:07:49	0
333	0.386	134	00:07:50	52
334	0.41300002	125	00:07:51	52
334	0	0	00:07:52	0
335	0.38300002	135	00:07:53	52
336	0.38300002	135	00:07:54	52
337	0.384	135	00:07:55	52
337	0	0	00:07:56	0
338	0.384	135	00:07:57	52
339	0.41300002	125	00:07:58	52
339	0	0	00:07:59	0
340	0.38300002	135	00:08:00	52
341	0.384	135	00:08:01	52
342	0.38300002	135	00:08:02	52
342	0	0	00:08:03	0
343	0.386	134	00:08:04	52
344	0.41300002	125	00:08:05	52
344	0	0	00:08:06	0
345	0.38300002	135	00:08:07	52
346	0.38300002	135	00:08:08	52
347	0.384	135	00:08:09	52
347	0	0	00:08:10	0
348	0.38300002	135	00:08:11	52
349	0.41500002	125	00:08:12	52
349	0	0	00:08:13	0
350	0.38300002	135	00:08:14	52
351	0.38500002	134	00:08:15	52
352	0.384	135	00:08:16	52
352	0	0	00:08:17	0
353	0.39400002	131	00:08:18	52
354	0.40300003	128	00:08:19	52
354	0	0	00:08:20	0
355	0.38300002	135	00:08:21	52
356	0.38500002	134	00:08:22	52
357	0.38300002	135	00:08:23	52
357	0	0	00:08:24	0
358	0.386	134	00:08:25	52
359	0.41700003	124	00:08:26	52
359	0	0	00:08:27	0
360	0.384	135	00:08:28	52
361	0.38500002	134	00:08:29	52
362	0.38300002	135	00:08:30	52
362	0	0	00:08:31	0
363	0.384	135	00:08:32	52
364	0.38700002	134	00:08:33	52
364	0	0	00:08:34	0
365	0.384	135	00:08:35	52
366	0.38300002	135	00:08:36	52
367	0.38300002	135	00:08:37	52
367	0	0	00:08:38	0
368	0.38900003	133	00:08:39	52
369	0.41400003	125	00:08:40	52
369	0	0	00:08:41	0
370	0.38300002	135	00:08:42	52
371	0.38300002	135	00:08:43	52
372	0.384	135	00:08:44	52
372	0	0	00:08:45	0
373	0.38700002	134	00:08:46	52
374	0.38700002	134	00:08:47	52
374	0	0	00:08:48	0
375	0.38300002	135	00:08:49	52
376	0.38500002	134	00:08:50	52
376	0	0	00:08:51	0
377	0.38300002	135	00:08:52	52
378	0.38500002	134	00:08:53	52
379	0.41300002	125	00:08:54	52
379	0	0	00:08:55	0
380	0.384	135	00:08:56	52
381	0.38300002	135	00:08:57	52
381	0	0	00:08:58	0
382	0.38300002	135	00:08:59	52
383	0.38700002	134	00:09:00	52
384	0.38700002	134	00:09:01	52
384	0	0	00:09:02	0
385	0.38300002	135	00:09:03	52
386	0.38300002	135	00:09:04	52
386	0	0	00:09:05	0
387	0.384	135	00:09:06	52
388	0.384	135	00:09:07	52
389	0.41300002	125	00:09:08	52
389	0	0	00:09:09	0
390	0.38300002	135	00:09:10	52
391	0.384	135	00:09:11	52
391	0	0	00:09:12	0
392	0.38300002	135	00:09:13	52
393	0.386	134	00:09:14	52
394	0.38700002	134	00:09:15	52
394	0	0	00:09:16	0
395	0.38300002	135	00:09:17	52
396	0.38300002	135	00:09:18	52
396	0	0	00:09:19	0
397	0.38300002	135	00:09:20	52
398	0.38500002	134	00:09:21	52
399	0.41400003	125	00:09:22	52
399	0	0	00:09:23	0
400	0.38300002	135	00:09:24	52
401	0.38300002	135	00:09:25	52
401	0	0	00:09:26	0
402	0.38300002	135	00:09:27	52
403	0.386	134	00:09:28	52
404	0.38700002	134	00:09:29	52
404	0	0	00:09:30	0
405	0.38300002	135	00:09:31	52
406	0.38300002	135	00:09:32	52
406	0	0	00:09:33	0
407	0.384	135	00:09:34	52
408	0.38500002	134	00:09:35	52
409	0.41300002	125	00:09:36	52
409	0	0	00:09:37	0
410	0.38300002	135	00:09:38	52
411	0.384	135	00:09:39	52
411	0	0	00:09:40	0
412	0.38300002	135	00:09:41	52
413	0.38300002	135	00:09:42	52
414	0.41400003	125	00:09:43	52
414	0	0	00:09:44	0
415	0.38300002	135	00:09:45	52
416	0.38300002	135	00:09:46	52
416	0	0	00:09:47	0
417	0.38300002	135	00:09:48	52
418	0.397	130	00:09:49	52
419	0.423	122	00:09:50	52
419	0	0	00:09:51	0
420	0.38300002	135	00:09:52	52
421	0.38300002	135	00:09:53	52
421	0	0	00:09:54	0
422	0.384	135	00:09:55	52
423	0.384	135	00:09:56	52
424	0.40300003	128	00:09:57	52
424	0	0	00:09:58	0
425	0.38300002	135	00:09:59	52
426	0.384	135	00:10:00	52
426	0	0	00:10:01	0
427	0.384	135	00:10:02	52
	166.150006	56801.6496		
Average	0.38911008	133.02494		

Measurement
Webserver

compr 1:1.0128 = 1.2%

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.384	132	00:00:04	51
2	0.43400002	116	00:00:05	51
2	0	0	00:00:06	0
3	0.39400002	128	00:00:07	51
4	0.38500002	131	00:00:08	51
4	0	0	00:00:09	0
5	0.384	132	00:00:10	51
6	0.384	132	00:00:11	51
7	0.45200002	112	00:00:12	51
7	0	0	00:00:13	0
8	0.38500002	131	00:00:14	51
9	0.384	132	00:00:15	51
9	0	0	00:00:16	0
10	0.384	132	00:00:17	51
11	0.38900003	130	00:00:18	51
12	0.384	132	00:00:19	51
12	0	0	00:00:20	0
13	0.384	132	00:00:21	51
14	0.39400002	128	00:00:22	51
14	0	0	00:00:23	0
15	0.38500002	131	00:00:24	51
16	0.39900002	127	00:00:25	51
17	0.39400002	128	00:00:26	51
17	0	0	00:00:27	0
18	0.384	132	00:00:28	51
19	0.38500002	131	00:00:29	51
19	0	0	00:00:30	0
20	0.384	132	00:00:31	51
21	0.39600003	128	00:00:32	51
22	0.38500002	131	00:00:33	51
22	0	0	00:00:34	0
23	0.384	132	00:00:35	51
24	0.384	132	00:00:36	51
24	0	0	00:00:37	0
25	0.384	132	00:00:38	51
26	0.38500002	131	00:00:39	51
27	0.404	125.05	00:00:40	51
27	0	0	00:00:41	0
28	0.384	132	00:00:42	51
29	0.38500002	131	00:00:43	51
29	0	0	00:00:44	0
30	0.384	132	00:00:45	51
31	0.38500002	131	00:00:46	51
32	0.41400003	122	00:00:47	51
32	0	0	00:00:48	0
33	0.38500002	131	00:00:49	51
34	0.386	131	00:00:50	51
34	0	0	00:00:51	0
35	0.384	132	00:00:52	51
36	0.384	132	00:00:53	51
36	0	0	00:00:54	0
37	0.41400003	122	00:00:55	51
38	0.384	132	00:00:56	51
39	0.384	132	00:00:57	51
39	0	0	00:00:58	0
40	0.384	132	00:00:59	51
41	0.38500002	131	00:01:00	51
41	0	0	00:01:01	0
42	0.40100002	126	00:01:02	51
43	0.402	126	00:01:03	51
44	0.384	132	00:01:04	51
44	0	0	00:01:05	0
45	0.38500002	131	00:01:06	51
46	0.384	132	00:01:07	51
46	0	0	00:01:08	0
47	0.384	132	00:01:09	51
48	0.39400002	128	00:01:10	51
49	0.38500002	131	00:01:11	51
49	0	0	00:01:12	0
50	0.384	132	00:01:13	51
51	0.384	132	00:01:14	51
51	0	0	00:01:15	0
52	0.38500002	131	00:01:16	51
53	0.41500002	122	00:01:17	51
54	0.384	132	00:01:18	51
54	0	0	00:01:19	0
55	0.384	132	00:01:20	51
56	0.38500002	131	00:01:21	51
56	0	0	00:01:22	0
57	0.38500002	131	00:01:23	51
58	0.404	125.05	00:01:24	51
59	0.384	132	00:01:25	51
59	0	0	00:01:26	0
60	0.38500002	131	00:01:27	51
61	0.38500002	131	00:01:28	51
61	0	0	00:01:29	0
62	0.386	131	00:01:30	51
63	0.38700002	131	00:01:31	51
64	0.384	132	00:01:32	51
64	0	0	00:01:33	0
65	0.384	132	00:01:34	51
66	0.384	132	00:01:35	51
66	0	0	00:01:36	0
67	0.38500002	131	00:01:37	51
68	0.41400003	122	00:01:38	51
69	0.384	132	00:01:39	51
69	0	0	00:01:40	0
70	0.384	132	00:01:41	51
71	0.38500002	131	00:01:42	51
71	0	0	00:01:43	0
72	0.38500002	131	00:01:44	51
73	0.404	125.05	00:01:45	51
74	0.38500002	131	00:01:46	51
74	0	0	00:01:47	0
75	0.38500002	131	00:01:48	51
76	0.384	132	00:01:49	51
76	0	0	00:01:50	0
77	0.384	132	00:01:51	51
78	0.41500002	122	00:01:52	51
79	0.384	132	00:01:53	51
79	0	0	00:01:54	0
80	0.384	132	00:01:55	51
81	0.38500002	131	00:01:56	51
81	0	0	00:01:57	0
82	0.38500002	131	00:01:58	51
83	0.41400003	122	00:01:59	51
84	0.384	132	00:02:00	51
84	0	0	00:02:01	0
85	0.38500002	131	00:02:02	51
86	0.384	132	00:02:03	51
86	0	0	00:02:04	0
87	0.384	132	00:02:05	51
88	0.402	126	00:02:06	51
88	0	0	00:02:07	0
89	0.395	128	00:02:08	51
90	0.384	132	00:02:09	51
91	0.384	132	00:02:10	51
91	0	0	00:02:11	0
92	0.38700002	131	00:02:12	51
93	0.404	125.05	00:02:13	51
93	0	0	00:02:14	0
94	0.38300002	132	00:02:15	51
95	0.384	132	00:02:16	51
96	0.38500002	131	00:02:17	51
96	0	0	00:02:18	0
97	0.38500002	131	00:02:19	51
98	0.404	125.05	00:02:20	51
98	0	0	00:02:21	0
99	0.38500002	131	00:02:22	51
100	0.38500002	131	00:02:23	51
101	0.384	132	00:02:24	51
101	0	0	00:02:25	0
102	0.388	130	00:02:26	51
103	0.41400003	122	00:02:27	51
103	0	0	00:02:28	0
104	0.384	132	00:02:29	51
105	0.384	132	00:02:30	51
106	0.384	132	00:02:31	51
106	0	0	00:02:32	0
107	0.38500002	131	00:02:33	51
108	0.404	125.05	00:02:34	51
108	0	0	00:02:35	0
109	0.384	132	00:02:36	51
110	0.38500002	131	00:02:37	51
111	0.38500002	131	00:02:38	51
111	0	0	00:02:39	0
112	0.386	131	00:02:40	51
113	0.41400003	122	00:02:41	51
113	0	0	00:02:42	0
114	0.38500002	131	00:02:43	51
115	0.384	132	00:02:44	51
116	0.384	132	00:02:45	51
116	0	0	00:02:46	0
117	0.38500002	131	00:02:47	51
118	0.388	130	00:02:48	51
118	0	0	00:02:49	0
119	0.384	132	00:02:50	51
120	0.384	132	00:02:51	51
121	0.38500002	131	00:02:52	51
121	0	0	00:02:53	0
122	0.388	130	00:02:54	51
123	0.41400003	122	00:02:55	51
123	0	0	00:02:56	0
124	0.384	132	00:02:57	51
125	0.38500002	131	00:02:58	51
126	0.384	132	00:02:59	51
126	0	0	00:03:00	0
127	0.384	132	00:03:01	51
128	0.38900003	130	00:03:02	51
128	0	0	00:03:03	0
129	0.384	132	00:03:04	51
130	0.384	132	00:03:05	51
131	0.384	132	00:03:06	51
131	0	0	00:03:07	0
132	0.38500002	131	00:03:08	51
133	0.432	117	00:03:09	51
133	0	0	00:03:10	0
134	0.384	132	00:03:11	51
135	0.384	132	00:03:12	51
136	0.38500002	131	00:03:13	51
136	0	0	00:03:14	0
137	0.43	117	00:03:15	51
138	0.384	132	00:03:16	51
138	0	0	00:03:17	0
139	0.43400002	116	00:03:18	51
140	0.38500002	131	00:03:19	51
141	0.384	132	00:03:20	51
141	0	0	00:03:21	0
142	0.384	132	00:03:22	51
143	0.38500002	131	00:03:23	51
143	0	0	00:03:24	0
144	0.418	121	00:03:25	51
145	0.384	132	00:03:26	51
145	0	0	00:03:27	0
146	0.384	132	00:03:28	51
147	0.38500002	131	00:03:29	51
148	0.38500002	131	00:03:30	51
148	0	0	00:03:31	0
149	0.425	119	00:03:32	51
150	0.384	132	00:03:33	51
150	0	0	00:03:34	0
151	0.38500002	131	00:03:35	51
152	0.384	132	00:03:36	51
153	0.384	132	00:03:37	51
153	0	0	00:03:38	0
154	0.41300002	122	00:03:39	51
155	0.38500002	131	00:03:40	51
155	0	0	00:03:41	0
156	0.384	132	00:03:42	51
157	0.384	132	00:03:43	51
158	0.388	130	00:03:44	51
158	0	0	00:03:45	0
159	0.41400003	122	00:03:46	51
160	0.384	132	00:03:47	51
160	0	0	00:03:48	0
161	0.384	132	00:03:49	51
162	0.38500002	131	00:03:50	51
163	0.384	132	00:03:51	51
163	0	0	00:03:52	0
164	0.41400003	122	00:03:53	51
165	0.38500002	131	00:03:54	51
165	0	0	00:03:55	0
166	0.384	132	00:03:56	51
167	0.384	132	00:03:57	51
168	0.384	132	00:03:58	51
168	0	0	00:03:59	0
169	0.41400003	122	00:04:00	51
170	0.64500004	78	00:04:01	51
170	0	0	00:04:02	0
171	0.40100002	126	00:04:03	51
172	0.43500003	116	00:04:04	51
172	0	0	00:04:05	0
173	0.393	128.55	00:04:06	51
174	0.384	132	00:04:07	51
175	0.384	132	00:04:08	51
175	0	0	00:04:09	0
176	0.38500002	131	00:04:10	51
177	0.41400003	122	00:04:11	51
177	0	0	00:04:12	0
178	0.384	132	00:04:13	51
179	0.384	132	00:04:14	51
180	0.38500002	131	00:04:15	51
180	0	0	00:04:16	0
181	0.384	132	00:04:17	51
182	0.41400003	122	00:04:18	51
182	0	0	00:04:19	0
183	0.384	132	00:04:20	51
184	0.384	132	00:04:21	51
184	0	0	00:04:22	0
185	0.384	132	00:04:23	51
186	0.384	132	00:04:24	51
187	0.38700002	131	00:04:25	51
187	0	0	00:04:26	0
188	0.384	132	00:04:27	51
189	0.384	132	00:04:28	51
189	0	0	00:04:29	0
190	0.38500002	131	00:04:30	51
191	0.38500002	131	00:04:31	51
192	0.41400003	122	00:04:32	51
192	0	0	00:04:33	0
193	0.384	132	00:04:34	51
194	0.38500002	131	00:04:35	51
194	0	0	00:04:36	0
195	0.384	132	00:04:37	51
196	0.384	132	00:04:38	51
197	0.41400003	122	00:04:39	51
197	0	0	00:04:40	0
198	0.38500002	131	00:04:41	51
199	0.384	132	00:04:42	51
199	0	0	00:04:43	0
200	0.384	132	00:04:44	51
201	0.384	132	00:04:45	51
202	0.404	125.05	00:04:46	51
202	0	0	00:04:47	0
203	0.384	132	00:04:48	51
204	0.384	132	00:04:49	51
204	0	0	00:04:50	0
205	0.39200002	129	00:04:51	51
206	0.384	132	00:04:52	51
207	0.404	125.05	00:04:53	51
207	0	0	00:04:54	0
208	0.38500002	131	00:04:55	51
209	0.38700002	131	00:04:56	51
209	0	0	00:04:57	0
210	0.384	132	00:04:58	51
211	0.42700002	118	00:04:59	51
212	0.38500002	131	00:05:00	51
212	0	0	00:05:01	0
213	0.47300002	107	00:05:02	51
214	0.384	132	00:05:03	51
214	0	0	00:05:04	0
215	0.384	132	00:05:05	51
216	0.384	132	00:05:06	51
216	0	0	00:05:07	0
217	0.432	117	00:05:08	51
218	0.384	132	00:05:09	51
219	0.384	132	00:05:10	51
219	0	0	00:05:11	0
220	0.384	132	00:05:12	51
221	0.38500002	131	00:05:13	51
222	0.38500002	131	00:05:14	51
222	0	0	00:05:15	0
223	0.41400003	122	00:05:16	51
224	0.384	132	00:05:17	51
224	0	0	00:05:18	0
225	0.38500002	131	00:05:19	51
226	0.384	132	00:05:20	51
226	0	0	00:05:21	0
227	0.41500002	122	00:05:22	51
228	0.38500002	131	00:05:23	51
229	0.384	132	00:05:24	51
229	0	0	00:05:25	0
230	0.384	132	00:05:26	51
231	0.384	132	00:05:27	51
231	0	0	00:05:28	0
232	0.38500002	131	00:05:29	51
233	0.41400003	122	00:05:30	51
234	0.384	132	00:05:31	51
234	0	0	00:05:32	0
235	0.38500002	131	00:05:33	51
236	0.384	132	00:05:34	51
236	0	0	00:05:35	0
237	0.384	132	00:05:36	51
238	0.41400003	122	00:05:37	51
239	0.38500002	131	00:05:38	51
239	0	0	00:05:39	0
240	0.384	132	00:05:40	51
241	0.384	132	00:05:41	51
241	0	0	00:05:42	0
242	0.384	132	00:05:43	51
243	0.404	125.05	00:05:44	51
244	0.384	132	00:05:45	51
244	0	0	00:05:46	0
245	0.384	132	00:05:47	51
246	0.38500002	131	00:05:48	51
246	0	0	00:05:49	0
247	0.38500002	131	00:05:50	51
248	0.41400003	122	00:05:51	51
249	0.384	132	00:05:52	51
249	0	0	00:05:53	0
250	0.38500002	131	00:05:54	51
251	0.384	132	00:05:55	51
251	0	0	00:05:56	0
252	0.384	132	00:05:57	51
253	0.41400003	122	00:05:58	51
254	0.38500002	131	00:05:59	51
254	0	0	00:06:00	0
255	0.384	132	00:06:01	51
256	0.4	126.3	00:06:02	51
256	0	0	00:06:03	0
257	0.4	126.3	00:06:04	51
258	0.39400002	128	00:06:05	51
259	0.384	132	00:06:06	51
259	0	0	00:06:07	0
260	0.384	132	00:06:08	51
261	0.384	132	00:06:09	51
261	0	0	00:06:10	0
262	0.38500002	131	00:06:11	51
263	0.41400003	122	00:06:12	51
264	0.384	132	00:06:13	51
264	0	0	00:06:14	0
265	0.38500002	131	00:06:15	51
266	0.384	132	00:06:16	51
266	0	0	00:06:17	0
267	0.384	132	00:06:18	51
268	0.41400003	122	00:06:19	51
269	0.38500002	131	00:06:20	51
269	0	0	00:06:21	0
270	0.384	132	00:06:22	51
271	0.384	132	00:06:23	51
271	0	0	00:06:24	0
272	0.384	132	00:06:25	51
273	0.40500003	125	00:06:26	51
274	0.384	132	00:06:27	51
274	0	0	00:06:28	0
275	0.384	132	00:06:29	51
276	0.38500002	131	00:06:30	51
276	0	0	00:06:31	0
277	0.402	126	00:06:32	51
278	0.41400003	122	00:06:33	51
279	0.384	132	00:06:34	51
279	0	0	00:06:35	0
280	0.38500002	131	00:06:36	51
281	0.384	132	00:06:37	51
281	0	0	00:06:38	0
282	0.384	132	00:06:39	51
283	0.41400003	122	00:06:40	51
284	0.38500002	131	00:06:41	51
284	0	0	00:06:42	0
285	0.39200002	129	00:06:43	51
286	0.384	132	00:06:44	51
286	0	0	00:06:45	0
287	0.384	132	00:06:46	51
288	0.404	125.05	00:06:47	51
289	0.386	131	00:06:48	51
289	0	0	00:06:49	0
290	0.384	132	00:06:50	51
291	0.38500002	131	00:06:51	51
291	0	0	00:06:52	0
292	0.38500002	131	00:06:53	51
293	0.41400003	122	00:06:54	51
294	0.384	132	00:06:55	51
294	0	0	00:06:56	0
295	0.395	128	00:06:57	51
296	0.384	132	00:06:58	51
296	0	0	00:06:59	0
297	0.384	132	00:07:00	51
298	0.40500003	125	00:07:01	51
299	0.384	132	00:07:02	51
299	0	0	00:07:03	0
300	0.384	132	00:07:04	51
301	0.384	132	00:07:05	51
301	0	0	00:07:06	0
302	0.38700002	131	00:07:07	51
303	0.41400003	122	00:07:08	51
304	0.384	132	00:07:09	51
304	0	0	00:07:10	0
305	0.38500002	131	00:07:11	51
306	0.384	132	00:07:12	51
306	0	0	00:07:13	0
307	0.384	132	00:07:14	51
308	0.41400003	122	00:07:15	51
308	0	0	00:07:16	0
309	0.38500002	131	00:07:17	51
310	0.384	132	00:07:18	51
311	0.384	132	00:07:19	51
311	0	0	00:07:20	0
312	0.384	132	00:07:21	51

313	0.40500003	125	00:07:22	51	353	0.384	132	00:08:18	51
313	0	0	00:07:23	0	353	0	0	00:08:19	0
314	0.384	132	00:07:24	51	354	0.38500002	131	00:08:20	51
315	0.384	132	00:07:25	51	355	0.384	132	00:08:21	51
316	0.38500002	131	00:07:26	51	356	0.384	132	00:08:22	51
316	0	0	00:07:27	0	356	0	0	00:08:23	0
317	0.38500002	131	00:07:28	51	357	0.38500002	131	00:08:24	51
318	0.41400003	122	00:07:29	51	358	0.41400003	122	00:08:25	51
318	0	0	00:07:30	0	358	0	0	00:08:26	0
319	0.384	132	00:07:31	51	359	0.384	132	00:08:27	51
320	0.38500002	131	00:07:32	51	360	0.384	132	00:08:28	51
321	0.384	132	00:07:33	51	361	0.38500002	131	00:08:29	51
321	0	0	00:07:34	0	361	0	0	00:08:30	0
322	0.384	132	00:07:35	51	362	0.41900003	121	00:08:31	51
323	0.41400003	122	00:07:36	51	363	0.384	132	00:08:32	51
323	0	0	00:07:37	0	363	0	0	00:08:33	0
324	0.38500002	131	00:07:38	51	364	0.384	132	00:08:34	51
325	0.384	132	00:07:39	51	365	0.38500002	131	00:08:35	51
326	0.384	132	00:07:40	51	366	0.384	132	00:08:36	51
326	0	0	00:07:41	0	366	0	0	00:08:37	0
327	0.38500002	131	00:07:42	51	367	0.384	132	00:08:38	51
328	0.418	121	00:07:43	51	368	0.41400003	122	00:08:39	51
328	0	0	00:07:44	0	368	0	0	00:08:40	0
329	0.39400002	128	00:07:45	51	369	0.38500002	131	00:08:41	51
330	0.384	132	00:07:46	51	370	0.38500002	131	00:08:42	51
331	0.38500002	131	00:07:47	51	371	0.384	132	00:08:43	51
331	0	0	00:07:48	0	371	0	0	00:08:44	0
332	0.395	128	00:07:49	51	372	0.416	121	00:08:45	51
333	0.384	132	00:07:50	51	373	0.384	132	00:08:46	51
333	0	0	00:07:51	0	373	0	0	00:08:47	0
334	0.38500002	131	00:07:52	51	374	0.39200002	129	00:08:48	51
335	0.384	132	00:07:53	51	375	0.384	132	00:08:49	51
336	0.384	132	00:07:54	51	376	0.38500002	131	00:08:50	51
336	0	0	00:07:55	0	376	0	0	00:08:51	0
337	0.384	132	00:07:56	51	377	0.384	132	00:08:52	51
338	0.41400003	122	00:07:57	51	378	0.404	125.05	00:08:53	51
338	0	0	00:07:58	0	378	0	0	00:08:54	0
339	0.384	132	00:07:59	51	379	0.38500002	131	00:08:55	51
340	0.384	132	00:08:00	51	380	0.38500002	131	00:08:56	51
341	0.38500002	131	00:08:01	51	381	0.384	132	00:08:57	51
341	0	0	00:08:02	0	381	0	0	00:08:58	0
342	0.43300003	117	00:08:03	51	382	0.384	132	00:08:59	51
343	0.38500002	131	00:08:04	51	383	0.41400003	122	00:09:00	51
343	0	0	00:08:05	0	383	0	0	00:09:01	0
344	0.384	132	00:08:06	51	384	0.54200006	93	00:09:02	51
345	0.384	132	00:08:07	51	385	0.43500003	116	00:09:03	51
346	0.38500002	131	00:08:08	51	385	0	0	00:09:04	0
346	0	0	00:08:09	0	386	0.384	132	00:09:05	51
347	0.38500002	131	00:08:10	51	387	0.384	132	00:09:06	51
348	0.41400003	122	00:08:11	51	388	0.384	132	00:09:07	51
348	0	0	00:08:12	0	388	0	0	00:09:08	0
349	0.384	132	00:08:13	51	389	0.393	128.55	00:09:09	51
350	0.38500002	131	00:08:14	51	390	0.41400003	122	00:09:10	51
351	0.384	132	00:08:15	51	390	0	0	00:09:11	0
351	0	0	00:08:16	0	391	0.384	132	00:09:12	51
352	0.407	124	00:08:17	51	392	0.38500002	131	00:09:13	51
392	0	0	00:09:14	0					
393	0.38500002	131	00:09:15	51					
394	0.384	132	00:09:16	51					
395	0.41400003	122	00:09:17	51					
395	0	0	00:09:18	0					
396	0.38500002	131	00:09:19	51					
397	0.38500002	131	00:09:20	51					
397	0	0	00:09:21	0					
398	0.384	132	00:09:22	51					
399	0.384	132	00:09:23	51					
400	0.41500002	122	00:09:24	51					
400	0	0	00:09:25	0					
401	0.384	132	00:09:26	51					
402	0.384	132	00:09:27	51					
402	0	0	00:09:28	0					
403	0.384	132	00:09:29	51					
404	0.38500002	131	00:09:30	51					
405	0.41400003	122	00:09:31	51					
405	0	0	00:09:32	0					
406	0.384	132	00:09:33	51					
407	0.386	131	00:09:34	51					
407	0	0	00:09:35	0					
408	0.384	132	00:09:36	51					
409	0.40800002	124	00:09:37	51					
410	0.384	132	00:09:38	51					
410	0	0	00:09:39	0					
411	0.38500002	131	00:09:40	51					
412	0.384	132	00:09:41	51					
412	0	0	00:09:42	0					
413	0.38300002	132	00:09:43	51					
414	0.38500002	131	00:09:44	51					
415	0.41400003	122	00:09:45	51					
415	0	0	00:09:46	0					
416	0.384	132	00:09:47	51					
417	0.38700002	131	00:09:48	51					
417	0	0	00:09:49	0					
418	0.38500002	131	00:09:50	51					
419	0.384	132	00:09:51	51					
420	0.404	125.05	00:09:52	51					
420	0	0	00:09:53	0					
421	0.38500002	131	00:09:54	51					
422	0.38500002	131	00:09:55	51					
422	0	0	00:09:56	0					
423	0.384	132	00:09:57	51					
424	0.384	132	00:09:58	51					
425	0.41500002	122	00:09:59	51					
425	0	0	00:10:00	0					
426	0.384	132	00:10:01	51					

Sum	166.856004	55049.6378							
Average	0.39168076	129.224502							

Measurements Web Server

compressed 1:1.5727 = 36%

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.37	85	00:00:04	32
2	0.29200003	108	00:00:05	32
2	0	0	00:00:06	0
3	0.264	120	00:00:07	32
4	0.30100003	105	00:00:08	32
5	0.26200002	121	00:00:09	32
6	0.29500002	107	00:00:10	32
6	0	0	00:00:11	0
7	0.26000002	122	00:00:12	32
8	0.30800003	103	00:00:13	32
9	0.28500003	111	00:00:14	32
9	0	0	00:00:15	0
10	0.264	120	00:00:16	32
11	0.273	116	00:00:17	32
12	0.26700002	118	00:00:18	32
13	0.261	121	00:00:19	32
13	0	0	00:00:20	0
14	0.264	120	00:00:21	32
15	0.26000002	122	00:00:22	32
16	0.261	121	00:00:23	32
17	0.259	122	00:00:24	32
17	0	0	00:00:25	0
18	0.293	108	00:00:26	32
19	0.27400002	115	00:00:27	32
20	0.259	122	00:00:28	32
21	0.256	124	00:00:29	32
21	0	0	00:00:30	0
22	0.259	122	00:00:31	32
23	0.256	124	00:00:32	32
24	0.26000002	122	00:00:33	32
24	0	0	00:00:34	0
25	0.305	104	00:00:35	32
26	0.286	111	00:00:36	32
27	0.25800002	123	00:00:37	32
28	0.26000002	122	00:00:38	32
28	0	0	00:00:39	0
29	0.25800002	123	00:00:40	32
30	0.26000002	122	00:00:41	32
31	0.259	122	00:00:42	32
32	0.26000002	122	00:00:43	32
32	0	0	00:00:44	0
33	0.257	123	00:00:45	32
34	0.26000002	122	00:00:46	32
35	0.25800002	123	00:00:47	32
36	0.29700002	107	00:00:48	32
36	0	0	00:00:49	0
37	0.25800002	123	00:00:50	32
38	0.261	121	00:00:51	32
39	0.25800002	123	00:00:52	32
39	0	0	00:00:53	0
40	0.26000002	122	00:00:54	32
41	0.261	121	00:00:55	32
42	0.31	102	00:00:56	32
43	0.28	113	00:00:57	32
43	0	0	00:00:58	0
44	0.293	108	00:00:59	32
45	0.26500002	119.37	00:01:00	32
46	0.26000002	122	00:01:01	32
46	0	0	00:01:02	0
47	0.284	111	00:01:03	32
48	0.26000002	122	00:01:04	32
49	0.257	123	00:01:05	32
50	0.26000002	122	00:01:06	32
50	0	0	00:01:07	0
51	0.259	122	00:01:08	32
52	0.259	122	00:01:09	32
53	0.257	123	00:01:10	32
54	0.259	122	00:01:11	32
54	0	0	00:01:12	0
55	0.305	104	00:01:13	32
56	0.26000002	122	00:01:14	32
57	0.25800002	123	00:01:15	32
58	0.26000002	122	00:01:16	32
58	0	0	00:01:17	0
59	0.257	123	00:01:18	32
60	0.26000002	122	00:01:19	32
61	0.27600002	115	00:01:20	32
61	0	0	00:01:21	0
62	0.3	105	00:01:22	32
63	0.25800002	123	00:01:23	32
64	0.264	120	00:01:24	32
65	0.259	122	00:01:25	32
65	0	0	00:01:26	0
66	0.259	122	00:01:27	32
67	0.25800002	123	00:01:28	32
68	0.26000002	122	00:01:29	32
69	0.26500002	119.37	00:01:30	32
69	0	0	00:01:31	0
70	0.26000002	122	00:01:32	32
71	0.25800002	123	00:01:33	32
72	0.26000002	122	00:01:34	32
73	0.25800002	123	00:01:35	32
73	0	0	00:01:36	0
74	0.30200002	105	00:01:37	32
75	0.25800002	123	00:01:38	32
76	0.259	122	00:01:39	32
77	0.25800002	123	00:01:40	32
77	0	0	00:01:41	0
78	0.26000002	122	00:01:42	32
79	0.25800002	123	00:01:43	32
80	0.31100002	102	00:01:44	32
80	0	0	00:01:45	0
81	0.284	111	00:01:46	32
82	0.263	120	00:01:47	32
83	0.259	122	00:01:48	32
84	0.26000002	122	00:01:49	32
84	0	0	00:01:50	0
85	0.25800002	123	00:01:51	32
86	0.259	122	00:01:52	32
87	0.25800002	123	00:01:53	32
88	0.261	121	00:01:54	32
88	0	0	00:01:55	0
89	0.259	122	00:01:56	32
90	0.264	120	00:01:57	32
91	0.30600002	103	00:01:58	32
92	0.259	122	00:01:59	32
92	0	0	00:02:00	0
93	0.58100003	54	00:02:01	32
94	0.26200002	121	00:02:02	32
94	0	0	00:02:03	0
95	0.259	122	00:02:04	32
96	0.26000002	122	00:02:05	32
97	0.257	123	00:02:06	32
98	0.26000002	122	00:02:07	32
98	0	0	00:02:08	0
99	0.25800002	123	00:02:09	32
100	0.259	122	00:02:10	32
101	0.26000002	122	00:02:11	32
102	0.259	122	00:02:12	32
102	0	0	00:02:13	0
103	0.25800002	123	00:02:14	32
104	0.259	122	00:02:15	32
105	0.257	123	00:02:16	32
106	0.259	122	00:02:17	32
106	0	0	00:02:18	0
107	0.282	112	00:02:19	32
108	0.26000002	122	00:02:20	32
109	0.26200002	121	00:02:21	32
109	0	0	00:02:22	0
110	0.26000002	122	00:02:23	32
111	0.25800002	123	00:02:24	32
112	0.26000002	122	00:02:25	32
113	0.30900002	102	00:02:26	32
113	0	0	00:02:27	0
114	0.26700002	118	00:02:28	32
115	0.257	123	00:02:29	32
116	0.259	122	00:02:30	32
117	0.256	124	00:02:31	32
117	0	0	00:02:32	0
118	0.25800002	123	00:02:33	32
119	0.256	124	00:02:34	32
120	0.259	122	00:02:35	32
121	0.256	124	00:02:36	32
121	0	0	00:02:37	0
122	0.26000002	122	00:02:38	32
123	0.257	123	00:02:39	32
124	0.259	122	00:02:40	32
125	0.25800002	123	00:02:41	32
125	0	0	00:02:42	0
126	0.259	122	00:02:43	32
127	0.25800002	123	00:02:44	32
128	0.294	108	00:02:45	32
128	0	0	00:02:46	0
129	0.257	123	00:02:47	32
130	0.287	110	00:02:48	32
131	0.25800002	123	00:02:49	32
132	0.31100002	102	00:02:50	32
132	0	0	00:02:51	0
133	0.27100003	117	00:02:52	32
134	0.26000002	122	00:02:53	32
135	0.25800002	123	00:02:54	32
136	0.259	122	00:02:55	32
136	0	0	00:02:56	0
137	0.263	120	00:02:57	32
138	0.259	122	00:02:58	32
139	0.259	122	00:02:59	32
140	0.26000002	122	00:03:00	32
140	0	0	00:03:01	0
141	0.298	106	00:03:02	32
142	0.25800002	123	00:03:03	32
143	0.257	123	00:03:04	32
143	0	0	00:03:05	0
144	0.26000002	122	00:03:06	32
145	0.257	123	00:03:07	32
146	0.259	122	00:03:08	32
147	0.30800003	103	00:03:09	32
147	0	0	00:03:10	0
148	0.29200003	108	00:03:11	32
149	0.25800002	123	00:03:12	32
150	0.259	122	00:03:13	32
151	0.25800002	123	00:03:14	32
151	0	0	00:03:15	0
152	0.26000002	122	00:03:16	32
153	0.25800002	123	00:03:17	32
154	0.261	121	00:03:18	32
154	0	0	00:03:19	0
155	0.25800002	123	00:03:20	32
156	0.259	122	00:03:21	32
157	0.26000002	122	00:03:22	32
158	0.29900002	106	00:03:23	32
158	0	0	00:03:24	0
159	0.25800002	123	00:03:25	32
160	0.259	122	00:03:26	32
161	0.261	121	00:03:27	32
162	0.26000002	122	00:03:28	32
162	0	0	00:03:29	0
163	0.29900002	106	00:03:30	32
164	0.26000002	122	00:03:31	32
165	0.27800003	114	00:03:32	32
166	0.26000002	122	00:03:33	32
166	0	0	00:03:34	0
167	0.25800002	123	00:03:35	32
168	0.259	122	00:03:36	32
169	0	0	00:03:38	0
170	0.259	122	00:03:39	32
171	0.25800002	123	00:03:40	32
172	0.259	122	00:03:41	32
173	0.25800002	123	00:03:42	32
173	0	0	00:03:43	0
174	0.259	122	00:03:44	32
175	0.30200002	105	00:03:45	32
176	0.26000002	122	00:03:46	32
177	0.25800002	123	00:03:47	32
177	0	0	00:03:48	0
178	0.26000002	122	00:03:49	32
179	0.305	104	00:03:50	32
180	0.289	109	00:03:51	32
180	0	0	00:03:52	0
181	0.25800002	123	00:03:53	32
182	0.26900002	118	00:03:54	32
183	0.25800002	123	00:03:55	32
184	0.26000002	122	00:03:56	32
184	0	0	00:03:57	0
185	0.26000002	122	00:03:58	32
186	0.259	122	00:03:59	32
187	0.25800002	123	00:04:00	32
188	0.30100003	105	00:04:01	32
188	0	0	00:04:02	0
189	0.25800002	123	00:04:03	32
190	0.25800002	123	00:04:04	32
191	0.257	123	00:04:05	32
192	0.26000002	122	00:04:06	32
192	0	0	00:04:07	0
193	0.259	122	00:04:08	32
194	0.307	103	00:04:09	32
195	0.275	115	00:04:10	32
195	0	0	00:04:11	0
196	0.26000002	122	00:04:12	32
197	0.25800002	123	00:04:13	32
198	0.259	122	00:04:14	32
199	0.257	123	00:04:15	32
199	0	0	00:04:16	0
200	0.26000002	122	00:04:17	32
201	0.25800002	123	00:04:18	32
202	0.26000002	122	00:04:19	32
203	0.303	104	00:04:20	32
203	0	0	00:04:21	0
204	0.26000002	122	00:04:22	32
205	0.25800002	123	00:04:23	32
206	0.26000002	122	00:04:24	32
206	0	0	00:04:25	0
207	0.31300002	101	00:04:26	32
208	0.284	111	00:04:27	32
209	0.25800002	123	00:04:28	32
210	0.26000002	122	00:04:29	32
210	0	0	00:04:30	0
211	0.261	121	00:04:31	32
212	0.26700002	118	00:04:32	32
213	0.25800002	123	00:04:33	32
214	0.26000002	122	00:04:34	32
214	0	0	00:04:35	0
215	0.25800002	123	00:04:36	32
216	0.259	122	00:04:37	32
217	0.25800002	123	00:04:38	32
218	0.294	108	00:04:39	32
218	0	0	00:04:40	0
219	0.25800002	123	00:04:41	32
220	0.26000002	122	00:04:42	32
221	0.25800002	123	00:04:43	32
222	0.259	122	00:04:44	32
222	0	0	00:04:45	0
223	0.25800002	123	00:04:46	32
224	0.30600002	103	00:04:47	32
225	0.28100002	113	00:04:48	32
225	0	0	00:04:49	0
226	0.26000002	122	00:04:50	32
227	0.259	122	00:04:51	32
228	0.259	122	00:04:52	32
229	0.25800002	123	00:04:53	32
229	0	0	00:04:54	0
230	0.26000002	122	00:04:55	32
231	0.25800002	123	00:04:56	32
232	0.259	122	00:04:57	32
233	0.25800002	123	00:04:58	32
233	0	0	00:04:59	0
234	0.26000002	122	00:05:00	32
235	0.453	69.83	00:05:01	32
236	0.259	122	00:05:02	32
236	0	0	00:05:03	0
237	0.25800002	123	00:05:04	32
238	0.259	122	00:05:05	32
239	0.25800002	123	00:05:06	32
240	0.305	104	00:05:07	32
240	0	0	00:05:08	0
241	0.25800002	123	00:05:09	32
242	0.26000002	122	00:05:10	32
243	0.25800002	123	00:05:11	32
243	0	0	00:05:12	0
244	0.31100002	102	00:05:13	32
245	0.28	113	00:05:14	32
246	0.261	121	00:05:15	32
247	0.26200002	121	00:05:16	32
247	0	0	00:05:17	0
248	0.26000002	122	00:05:18	32
249	0.25800002	123	00:05:19	32
250	0.259	122	00:05:20	32
251	0.26000002	122	00:05:21	32
251	0	0	00:05:22	0
252	0.261	121	00:05:23	32
253	0.25800002	123	00:05:24	32
254	0.259	122	00:05:25	32
254	0	0	00:05:26	0
255	0.30100003	105	00:05:27	32
256	0.264	120	00:05:28	32
257	0.259	122	00:05:29	32
258	0.26000002	122	00:05:30	32
258	0	0	00:05:31	0
259	0.31100002	102	00:05:32	32
260	0.27600002	115	00:05:33	32
261	0.25800002	123	00:05:34	32
262	0.26000002	122	00:05:35	32
262	0	0	00:05:36	0
263	0.25800002	123	00:05:37	32
264	0.259	122	00:05:38	32
265	0.257	123	00:05:39	32
266	0.259	122	00:05:40	32
266	0	0	00:05:41	0
267	0.259	122	00:05:42	32
268	0.259	122	00:05:43	32
269	0.25800002	123	00:05:44	32
269	0	0	00:05:45	0
270	0.289	109	00:05:46	32
271	0.25800002	123	00:05:47	32
272	0.26700002	118	00:05:48	32
273	0.25800002	123	00:05:49	32
273	0	0	00:05:50	0
274	0.31	102	00:05:51	32
275	0.27	117	00:05:52	32
276	0.26000002	122	00:05:53	32
277	0.259	122	00:05:54	32
277	0	0	00:05:55	0
278	0.261	121	00:05:56	32
279	0.25800002	123	00:05:57	32
280	0.26000002	122	00:05:58	32
281	0.25800002	123	00:05:59	32
281	0	0	00:06:00	0
282	0.261	121	00:06:01	32
283	0.257	123	00:06:02	32
284	0.26000002	122	00:06:03	32
284	0	0	00:06:04	0
285	0.303	104	00:06:05	32
286	0.25800002	123	00:06:06	32
287	0.256	124	00:06:07	32
288	0.261	121	00:06:08	32
288	0	0	00:06:09	0
289	0.31	102	00:06:10	32
290	0.277	114	00:06:11	32
291	0.257	123	00:06:12	32
292	0.26000002	122	00:06:13	32
292	0	0	00:06:14	0
293	0.261	121	00:06:15	32
294	0.259	122	00:06:16	32
295	0.25800002	123	00:06:17	32
295	0	0	00:06:18	0
296	0.26000002	122	00:06:19	32
297	0.257	123	00:06:20	32
298	0.259	122	00:06:21	32
299	0.25800002	123	00:06:22	32
299	0	0	00:06:23	0
300	0.26000002	122	00:06:24	32
301	0.257	123	00:06:25	32
302	0.305	104	00:06:26	32
303	0.257	123	00:06:27	32
303	0	0	00:06:28	0
304	0.26000002	122	00:06:29	32
305	0.25800002	123	00:06:30	32
306	0.259	122	00:06:31	32
307	0.25800002	123	00:06:32	32
307	0	0	00:06:33	0
308	0.312	101	00:06:34	32
309	0.268	118	00:06:35	32
310	0.259	122	00:06:36	32
310	0	0	00:06:37	0
311	0.25800002	123	00:06:38	32
312	0.26000002	122	00:06:39	32
313	0.25800002	123	00:06:40	32
314	0.26000002	122	00:06:41	32
314	0	0	00:06:42	0
315	0.25800002	123	00:06:43	32
316	0.26000002	122	00:06:44	32
317	0.25800002	123	00:06:45	32
318	0.26000002	122	00:06:46	32
318	0	0	00:06:47	0
319	0.25800002	123	00:06:48	32
320	0.259	122	00:06:49	32
321	0.287	110	00:06:50	32
322	0.259	122	00:06:51	32
322	0	0	00:06:52	0
323	0.257	123	00:06:53	32
324	0.26000002	122	00:06:54	32
325	0.25800002	123	00:06:55	32
326	0.26000002	122	00:06:56	32
326	0	0	00:06:57	0
327	0.30900002	102	00:06:58	32
328	0.28500003	111	00:06:59	32
329	0.25800002	123	00:07:00	32
329	0	0	00:07:01	0
330	0.259	122	00:07:02	32
331	0.257	123	00:07:03	32
332	0.261	121	00:07:04	32
333	0.257	123	00:07:05	32
333	0	0	00:07:06	0
334	0.26000002	122	00:07:07	32
335	0.25800002	123	00:07:08	32
336	0.259	122	00:07:09	32
337	0.25800002	123	00:07:10	32
337	0	0	00:07:11	0
338	0.259	122	00:07:12	32
339	0.257	123	00:07:13	32
340	0.28500003	111	00:07:14	32
341	0.256	124	00:07:15	32
341	0	0	00:07:16	0
342	0.25800002	123	00:07:17	32
343	0.25800002	123	00:07:18	32
344	0.259	122	00:07:19	32
345	0.25800002	123	00:07:20	32
345	0	0	00:07:21	0

346	0.277	114	00:07:22	32	390	0	0	00:08:18	0
347	0.293	108	00:07:23	32	391	0.25800002	123	00:08:19	32
348	0.26000002	122	00:07:24	32	392	0.259	122	00:08:20	32
348	0	0	00:07:25	0	393	0.28800002	110	00:08:21	32
349	0.27100003	117	00:07:26	32	393	0	0	00:08:22	0
350	0.26000002	122	00:07:27	32	394	0.26500002	119.37	00:08:23	32
351	0.25800002	123	00:07:28	32	395	0.259	122	00:08:24	32
352	0.26000002	122	00:07:29	32	396	0.26000002	122	00:08:25	32
352	0	0	00:07:30	0	397	0.257	123	00:08:26	32
353	0.257	123	00:07:31	32	397	0	0	00:08:27	0
354	0.26000002	122	00:07:32	32	398	0.26000002	122	00:08:28	32
355	0.25800002	123	00:07:33	32	399	0.31	102	00:08:29	32
356	0.26700002	118	00:07:34	32	400	0.26700002	118	00:08:30	32
356	0	0	00:07:35	0	401	0.257	123	00:08:31	32
357	0.305	104	00:07:36	32	401	0	0	00:08:32	0
358	0.26000002	122	00:07:37	32	402	0.261	121	00:08:33	32
359	0.257	123	00:07:38	32	403	0.25800002	123	00:08:34	32
360	0.26000002	122	00:07:39	32	404	0.26000002	122	00:08:35	32
360	0	0	00:07:40	0	405	0.259	122	00:08:36	32
361	0.25800002	123	00:07:41	32	405	0	0	00:08:37	0
362	0.26000002	122	00:07:42	32	406	0.261	121	00:08:38	32
363	0.30400002	104	00:07:43	32	407	0.25800002	123	00:08:39	32
363	0	0	00:07:44	0	408	0.26000002	122	00:08:40	32
364	0.282	112	00:07:45	32	409	0.25800002	123	00:08:41	32
365	0.25800002	123	00:07:46	32	409	0	0	00:08:42	0
366	0.263	120	00:07:47	32	410	0.26000002	122	00:08:43	32
367	0.25800002	123	00:07:48	32	411	0.25800002	123	00:08:44	32
367	0	0	00:07:49	0	412	0.30400002	104	00:08:45	32
368	0.259	122	00:07:50	32	412	0	0	00:08:46	0
369	0.25800002	123	00:07:51	32	413	0.259	122	00:08:47	32
370	0.259	122	00:07:52	32	414	0.259	122	00:08:48	32
371	0.25800002	123	00:07:53	32	415	0.25800002	123	00:08:49	32
371	0	0	00:07:54	0	416	0.28	113	00:08:50	32
372	0.26000002	122	00:07:55	32	416	0	0	00:08:51	0
373	0.26200002	121	00:07:56	32	417	0.289	109	00:08:52	32
374	0.30600002	103	00:07:57	32	418	0.26000002	122	00:08:53	32
375	0.25800002	123	00:07:58	32	419	0.268	118	00:08:54	32
375	0	0	00:07:59	0	420	0.26000002	122	00:08:55	32
376	0.26000002	122	00:08:00	32	420	0	0	00:08:56	0
377	0.25800002	123	00:08:01	32	421	0.25800002	123	00:08:57	32
378	0.261	121	00:08:02	32	422	0.259	122	00:08:58	32
378	0	0	00:08:03	0	423	0.25800002	123	00:08:59	32
379	0.259	122	00:08:04	32	424	0.259	122	00:09:00	32
380	0.31	102	00:08:05	32	424	0	0	00:09:01	0
381	0.279	113	00:08:06	32	425	0.25800002	123	00:09:02	32
382	0.26000002	122	00:08:07	32	426	0.26000002	122	00:09:03	32
382	0	0	00:08:08	0	427	0.25800002	123	00:09:04	32
383	0.257	123	00:08:09	32	427	0	0	00:09:05	0
384	0.26000002	122	00:08:10	32	428	0.26000002	122	00:09:06	32
385	0.25800002	123	00:08:11	32	429	0.25800002	123	00:09:07	32
386	0.259	122	00:08:12	32	430	0.259	122	00:09:08	32
386	0	0	00:08:13	0	431	0.286	111	00:09:09	32
387	0.257	123	00:08:14	32	431	0	0	00:09:10	0
388	0.26000002	122	00:08:15	32	432	0.26000002	122	00:09:11	32
389	0.257	123	00:08:16	32	433	0.257	123	00:09:12	32
390	0.259	122	00:08:17	32	434	0.259	122	00:09:13	32
435	0.25800002	123	00:09:14	32					
435	0	0	00:09:15	0					
436	0.261	121	00:09:16	32					
437	0.31100002	102	00:09:17	32					
438	0.28800002	110	00:09:18	32					
439	0.25800002	123	00:09:19	32					
439	0	0	00:09:20	0					
440	0.26000002	122	00:09:21	32					
441	0.257	123	00:09:22	32					
442	0.26000002	122	00:09:23	32					
442	0	0	00:09:24	0					
443	0.25800002	123	00:09:25	32					
444	0.26000002	122	00:09:26	32					
445	0.257	123	00:09:27	32					
446	0.259	122	00:09:28	32					
446	0	0	00:09:29	0					
447	0.257	123	00:09:30	32					
448	0.307	103	00:09:31	32					
449	0.25800002	123	00:09:32	32					
450	0.259	122	00:09:33	32					
450	0	0	00:09:34	0					
451	0.259	122	00:09:35	32					
452	0.26000002	122	00:09:36	32					
453	0.25800002	123	00:09:37	32					
454	0.307	103	00:09:38	32					
454	0	0	00:09:39	0					
455	0.28	113	00:09:40	32					
456	0.26000002	122	00:09:41	32					
457	0.257	123	00:09:42	32					
457	0	0	00:09:43	0					
458	0.26000002	122	00:09:44	32					
459	0.25800002	123	00:09:45	32					
460	0.26000002	122	00:09:46	32					
461	0.25800002	123	00:09:47	32					
461	0	0	00:09:48	0					
462	0.261	121	00:09:49	32					
463	0.30800003	103	00:09:50	32					
464	0.26000002	122	00:09:51	32					
465	0.25800002	123	00:09:52	32					
465	0	0	00:09:53	0					
466	0.26200002	121	00:09:54	32					
467	0.25800002	123	00:09:55	32					
468	0.263	120	00:09:56	32					
469	0.305	104	00:09:57	32					
469	0	0	00:09:58	0					
470	0.282	112	00:09:59	32					
471	0.25800002	123	00:10:00	32					
471	0	0	00:10:01	0					
472	0.52000004	61	00:10:02	32					
Sum	126.342006	56116.2301							
Average	0.26824205	118.890318							

Measurements
Webserver

compr 1:2.23762 = 55.31%

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.18800001	125	00:00:04	24
2	0.18800001	125	00:00:05	24
3	0.208	113.12	00:00:06	24
4	0.187	126	00:00:07	24
4	0	0	00:00:08	0
5	0.20700002	114	00:00:09	24
6	0.19700001	119	00:00:10	24
7	0.18900001	124	00:00:11	24
8	0.18800001	125	00:00:12	24
9	0.18800001	125	00:00:13	24
9	0	0	00:00:14	0
10	0.20700002	114	00:00:15	24
11	0.18800001	125	00:00:16	24
12	0.18800001	125	00:00:17	24
13	0.307	77	00:00:18	24
13	0	0	00:00:19	0
14	0.187	126	00:00:20	24
15	0.187	126	00:00:21	24
16	0.187	126	00:00:22	24
17	0.22700001	104	00:00:23	24
18	0.177	133	00:00:24	24
18	0	0	00:00:25	0
19	0.19000001	124	00:00:26	24
20	0.18800001	125	00:00:27	24
21	0.19700001	119	00:00:28	24
22	0.208	113.12	00:00:29	24
23	0.18800001	125	00:00:30	24
23	0	0	00:00:31	0
24	0.18800001	125	00:00:32	24
25	0.18800001	125	00:00:33	24
26	0.18800001	125	00:00:34	24
27	0.19800001	119	00:00:35	24
28	0.18800001	125	00:00:36	24
28	0	0	00:00:37	0
29	0.18800001	125	00:00:38	24
30	0.178	132	00:00:39	24
31	0.20400001	115	00:00:40	24
32	0.18100001	130	00:00:41	24
33	0.187	126	00:00:42	24
33	0	0	00:00:43	0
34	0.19500001	121	00:00:44	24
35	0.187	126	00:00:45	24
36	0.19700001	119	00:00:46	24
37	0.18800001	125	00:00:47	24
38	0.178	132	00:00:48	24
38	0	0	00:00:49	0
39	0.19700001	119	00:00:50	24
40	0.18800001	125	00:00:51	24
41	0.19700001	119	00:00:52	24
42	0.208	113.12	00:00:53	24
43	0.18800001	125	00:00:54	24
43	0	0	00:00:55	0
44	0.18800001	125	00:00:56	24
45	0.18800001	125	00:00:57	24
46	0.178	132	00:00:58	24
47	0.208	113.12	00:00:59	24
48	0.18800001	125	00:01:00	24
48	0	0	00:01:01	0
49	0.19900002	118	00:01:02	24
50	0.19600001	120	00:01:03	24
51	0.18800001	125	00:01:04	24
52	0.208	113.12	00:01:05	24
53	0.18800001	125	00:01:06	24
53	0	0	00:01:07	0
54	0.178	132	00:01:08	24
55	0.18800001	125	00:01:09	24
56	0.18800001	125	00:01:10	24
57	0.208	113.12	00:01:11	24
57	0	0	00:01:12	0
58	0.187	126	00:01:13	24
59	0.18800001	125	00:01:14	24
60	0.18800001	125	00:01:15	24
61	0.187	126	00:01:16	24
62	0.19700001	119	00:01:17	24
62	0	0	00:01:18	0
63	0.187	126	00:01:19	24
64	0.187	126	00:01:20	24
65	0.18400002	128	00:01:21	24
66	0.178	132	00:01:22	24
67	0.19600001	120	00:01:23	24
67	0	0	00:01:24	0
68	0.187	126	00:01:25	24
69	0.19700001	119	00:01:26	24
70	0.20700002	114	00:01:27	24
71	0.18800001	125	00:01:28	24
72	0.18800001	125	00:01:29	24
72	0	0	00:01:30	0
73	0.18800001	125	00:01:31	24
74	0.178	132	00:01:32	24
75	0.18800001	125	00:01:33	24
76	0.202	116.48	00:01:34	24
77	0.202	116.48	00:01:35	24
77	0	0	00:01:36	0
78	0.178	132	00:01:37	24
79	0.18800001	125	00:01:38	24
80	0.18800001	125	00:01:39	24
81	0.18800001	125	00:01:40	24
82	0.20700002	114	00:01:41	24
82	0	0	00:01:42	0
83	0.19600001	120	00:01:43	24
84	0.186	127	00:01:44	24
85	0.185	127	00:01:45	24
86	0.216	109	00:01:46	24
87	0.187	126	00:01:47	24
87	0	0	00:01:48	0
88	0.18900001	124	00:01:49	24
89	0.19600001	120	00:01:50	24
90	0.18800001	125	00:01:51	24
91	0.18800001	125	00:01:52	24
92	0.18800001	125	00:01:53	24
92	0	0	00:01:54	0
93	0.18800001	125	00:01:55	24
94	0.177	133	00:01:56	24
95	0.18800001	125	00:01:57	24
96	0.18800001	125	00:01:58	24
97	0.208	113.12	00:01:59	24
97	0	0	00:02:00	0
98	0.178	132	00:02:01	24
99	0.18800001	125	00:02:02	24
100	0.187	126	00:02:03	24
101	0.21900001	107	00:02:04	24
102	0.177	133	00:02:05	24
102	0	0	00:02:06	0
103	0.187	126	00:02:07	24
104	0.187	126	00:02:08	24
105	0.187	126	00:02:09	24
106	0.18800001	125	00:02:10	24
107	0.208	113.12	00:02:11	24
107	0	0	00:02:12	0
108	0.18800001	125	00:02:13	24
109	0.18800001	125	00:02:14	24
110	0.18800001	125	00:02:15	24
111	0.18800001	125	00:02:16	24
112	0.18800001	125	00:02:17	24
112	0	0	00:02:18	0
113	0.186	127	00:02:19	24
114	0.187	126	00:02:20	24
115	0.18800001	125	00:02:21	24
116	0.18800001	125	00:02:22	24
117	0.19700001	119	00:02:23	24
117	0	0	00:02:24	0
118	0.187	126	00:02:25	24
119	0.19600001	120	00:02:26	24
120	0.19600001	120	00:02:27	24
121	0.187	126	00:02:28	24
122	0.19700001	119	00:02:29	24
122	0	0	00:02:30	0
123	0.187	126	00:02:31	24
124	0.187	126	00:02:32	24
125	0.187	126	00:02:33	24
126	0.187	126	00:02:34	24
127	0.202	116.48	00:02:35	24
127	0	0	00:02:36	0
128	0.187	126	00:02:37	24
129	0.18800001	125	00:02:38	24
130	0.178	132	00:02:39	24
131	0.18800001	125	00:02:40	24
132	0.21800001	108	00:02:41	24
132	0	0	00:02:42	0
133	0.187	126	00:02:43	24
134	0.178	132	00:02:44	24
135	0.19700001	119	00:02:45	24
136	0.187	126	00:02:46	24
137	0.19700001	119	00:02:47	24
137	0	0	00:02:48	0
138	0.187	126	00:02:49	24
139	0.19900002	118	00:02:50	24
140	0.187	126	00:02:51	24
141	0.185	127	00:02:52	24
142	0.185	127	00:02:53	24
142	0	0	00:02:54	0
143	0.187	126	00:02:55	24
144	0.187	126	00:02:56	24
145	0.187	126	00:02:57	24
146	0.18800001	125	00:02:58	24
147	0.19600001	120	00:02:59	24
147	0	0	00:03:00	0
148	0.187	126	00:03:01	24
149	0.19900002	118	00:03:02	24
150	0.193	122	00:03:03	24
151	0.187	126	00:03:04	24
151	0	0	00:03:05	0
152	0.20700002	114	00:03:06	24
153	0.187	126	00:03:07	24
154	0.177	133	00:03:08	24
155	0.187	126	00:03:09	24
156	0.18800001	125	00:03:10	24
156	0	0	00:03:11	0
157	0.19800001	119	00:03:12	24
158	0.178	132	00:03:13	24
159	0.18800001	125	00:03:14	24
160	0.18800001	125	00:03:15	24
161	0.18800001	125	00:03:16	24
161	0	0	00:03:17	0
162	0.208	113.12	00:03:18	24
163	0.19700001	119	00:03:19	24
164	0.187	126	00:03:20	24
165	0.19700001	119	00:03:21	24
166	0.187	126	00:03:22	24
166	0	0	00:03:23	0
167	0.19700001	119	00:03:24	24
168	0.18800001	125	00:03:25	24
169	0.18800001	125	00:03:26	24
170	0.178	132	00:03:27	24
171	0.18800001	125	00:03:28	24
171	0	0	00:03:29	0
172	0.20700002	114	00:03:30	24
173	0.187	126	00:03:31	24
174	0.186	127	00:03:32	24
175	0.19600001	120	00:03:33	24
176	0.187	126	00:03:34	24
176	0	0	00:03:35	0
177	0.19700001	119	00:03:36	24
178	0.178	132	00:03:37	24
179	0.18800001	125	00:03:38	24
180	0.18800001	125	00:03:39	24
181	0.18800001	125	00:03:40	24
181	0	0	00:03:41	0
182	0.208	113.12	00:03:42	24
183	0.18800001	125	00:03:43	24
184	0.18800001	125	00:03:44	24
185	0.18800001	125	00:03:45	24
186	0.178	132	00:03:46	24
186	0	0	00:03:47	0
187	0.208	113.12	00:03:48	24
188	0.19600001	120	00:03:49	24
189	0.18800001	125	00:03:50	24
190	0.178	132	00:03:51	24
191	0.19700001	119	00:03:52	24
191	0	0	00:03:53	0
192	0.18800001	125	00:03:54	24
193	0.18800001	125	00:03:55	24
194	0.187	126	00:03:56	24
195	0.19600001	120	00:03:57	24
196	0.186	127	00:03:58	24
196	0	0	00:03:59	0
197	0.185	127	00:04:00	24
198	0.177	133	00:04:01	24
199	0.19600001	120	00:04:02	24
200	0.187	126	00:04:03	24
201	0.186	127	00:04:04	24
201	0	0	00:04:05	0
202	0.22800002	103	00:04:06	24
203	0.18800001	125	00:04:07	24
204	0.18800001	125	00:04:08	24
205	0.18800001	125	00:04:09	24
206	0.178	132	00:04:10	24
206	0	0	00:04:11	0
207	0.238	99	00:04:12	24
208	0.18800001	125	00:04:13	24
209	0.18800001	125	00:04:14	24
210	0.178	132	00:04:15	24
211	0.18800001	125	00:04:16	24
211	0	0	00:04:17	0
212	0.20700002	114	00:04:18	24
213	0.187	126	00:04:19	24
214	0.177	133	00:04:20	24
215	0.187	126	00:04:21	24
216	0.18800001	125	00:04:22	24
216	0	0	00:04:23	0
217	0.21800001	108	00:04:24	24
218	0.178	132	00:04:25	24
219	0.187	126	00:04:26	24
220	0.18800001	125	00:04:27	24
221	0.18800001	125	00:04:28	24
221	0	0	00:04:29	0
222	0.187	126	00:04:30	24
223	0.18800001	125	00:04:31	24
224	0.18800001	125	00:04:32	24
225	0.18800001	125	00:04:33	24
226	0.178	132	00:04:34	24
226	0	0	00:04:35	0
227	0.21800001	108	00:04:36	24
228	0.18800001	125	00:04:37	24
229	0.18800001	125	00:04:38	24
230	0.178	132	00:04:39	24
231	0.18800001	125	00:04:40	24
231	0	0	00:04:41	0
232	0.208	113.12	00:04:42	24
233	0.187	126	00:04:43	24
234	0.177	133	00:04:44	24
235	0.187	126	00:04:45	24
235	0	0	00:04:46	0
236	0.187	126	00:04:47	24
237	0.21700001	108	00:04:48	24
238	0.20600002	114	00:04:49	24
239	0.187	126	00:04:50	24
240	0.18800001	125	00:04:51	24
240	0	0	00:04:52	0
241	0.18800001	125	00:04:53	24
242	0.178	132	00:04:54	24
243	0.18800001	125	00:04:55	24
244	0.18800001	125	00:04:56	24
245	0.18800001	125	00:04:57	24
245	0	0	00:04:58	0
246	0.18800001	125	00:04:59	24
247	0.208	113.12	00:05:00	24
248	0.18900001	124	00:05:01	24
249	0.19800001	119	00:05:02	24
250	0.19600001	120	00:05:03	24
250	0	0	00:05:04	0
251	0.20400001	115	00:05:05	24
252	0.192	123	00:05:06	24
253	0.19700001	119	00:05:07	24
254	0.178	132	00:05:08	24
255	0.18800001	125	00:05:09	24
255	0	0	00:05:10	0
256	0.18800001	125	00:05:11	24
257	0.238	99	00:05:12	24
258	0.178	132	00:05:13	24
259	0.18800001	125	00:05:14	24
260	0.18800001	125	00:05:15	24
260	0	0	00:05:16	0
261	0.19700001	119	00:05:17	24
262	0.20700002	114	00:05:18	24
263	0.187	126	00:05:19	24
264	0.18800001	125	00:05:20	24
265	0.187	126	00:05:21	24
265	0	0	00:05:22	0
266	0.178	132	00:05:23	24
267	0.20700002	114	00:05:24	24
268	0.187	126	00:05:25	24
269	0.187	126	00:05:26	24
270	0.177	133	00:05:27	24
270	0	0	00:05:28	0
271	0.18800001	125	00:05:29	24
272	0.20700002	114	00:05:30	24
273	0.19700001	119	00:05:31	24
274	0.187	126	00:05:32	24
275	0.18800001	125	00:05:33	24
275	0	0	00:05:34	0
276	0.18800001	125	00:05:35	24
277	0.18800001	125	00:05:36	24
278	0.177	133	00:05:37	24
279	0.18800001	125	00:05:38	24
280	0.18800001	125	00:05:39	24
280	0	0	00:05:40	0
281	0.18800001	125	00:05:41	24
282	0.208	113.12	00:05:42	24
283	0.18800001	125	00:05:43	24
284	0.187	126	00:05:44	24
285	0.185	127	00:05:45	24
285	0	0	00:05:46	0
286	0.177	133	00:05:47	24
287	0.20700002	114	00:05:48	24
288	0.187	126	00:05:49	24
289	0.187	126	00:05:50	24
290	0.177	133	00:05:51	24
290	0	0	00:05:52	0
291	0.19700001	119	00:05:53	24
292	0.19600001	120	00:05:54	24
293	0.187	126	00:05:55	24
294	0.178	132	00:05:56	24
295	0.187	126	00:05:57	24
295	0	0	00:05:58	0
296	0.194	121	00:05:59	24
297	0.19700001	119	00:06:00	24
298	0.187	126	00:06:01	24
299	0.18800001	125	00:06:02	24
300	0.18800001	125	00:06:03	24
300	0	0	00:06:04	0
301	0.18800001	125	00:06:05	24
302	0.178	132	00:06:06	24
303	0.18800001	125	00:06:07	24
304	0.18800001	125	00:06:08	24
305	0.187	126	00:06:09	24
305	0	0	00:06:10	0
306	0.178	132	00:06:11	24
307	0.18800001	125	00:06:12	24
308	0.18800001	125	00:06:13	24
309	0.18800001	125	00:06:14	24
310	0.19700001	119	00:06:15	24
310	0	0	00:06:16	0
311	0.187	126	00:06:17	24
312	0.187	126	00:06:18	24
313	0.187	126	00:06:19	24
314	0.178	132	00:06:20	24
315	0.20700002	114	00:06:21	24
315	0	0	00:06:22	0
316	0.187	126	00:06:23	24
317	0.187	126	00:06:24	24
318	0.187	126	00:06:25	24
319	0.19700001	119	00:06:26	24
320	0.20700002	114	00:06:27	24
320	0	0	00:06:28	0
321	0.18800001	125	00:06:29	24
322	0.178	132	00:06:30	24
323	0.18800001	125	00:06:31	24
324	0.18800001	125	00:06:32	24
325	0.208	113.12	00:06:33	24
325	0	0	00:06:34	0
326	0.178	132	00:06:35	24
327	0.18800001	125	00:06:36	24
328	0.18800001	125	00:06:37	24
329	0.18800001	125	00:06:38	24
330	0.208	113.12	00:06:39	24
330	0	0	00:06:40	0
331	0.18800001	125	00:06:41	24
332	0.18800001	125	00:06:42	24
333	0.187	126	00:06:43	24
334	0.186	127	00:06:44	24
334	0	0	00:06:45	0
335	0.20700002	114	00:06:46	24
336	0.187	126	00:06:47	24
337	0.187	126	00:06:48	24
338	0.177	133	00:06:49	24
339	0.187	126	00:06:50	24
339	0	0	00:06:51	0
340	0.18100001	130	00:06:52	24
341	0.18800001	125	00:06:53	24
342	0.2	118	00:06:54	24
343	0.18800001	125	00:06:55	24
344	0.18800001	125	00:06:56	24
344	0	0	00:06:57	0
345	0.18800001	125	00:06:58	24
346	0.178	132	00:06:59	24
347	0.18800001	125	00:07:00	24
348	0.18800001	125	00:07:01	24
349	0.19700001	119	00:07:02	24
349	0	0	00:07:03	0
350	0.194	121	00:07:04	24
351	0.187	126	00:07:05	24
352	0.187	126	00:07:06	24
353	0.187	126	00:07:07	24
354	0.177	133	00:07:08	24
354	0	0	00:07:09	0
355	0.18800001	125	00:07:10	24
356	0.18800001	125	00:07:11	24
357	0.18800001	125	00:07:12	24
358	0.21700001	108	00:07:13	24
359	0.18800001	125	00:07:14	24
359	0	0	00:07:15	0
360	0.18800001	125	00:07:16	24
361	0.186	127	00:07:17	24
362	0.187	126	00:07:18	24
363	0.18800001	125	00:07:19	24
364	0.19800001	119	00:07:20	24
364	0	0	00:07:21	0

365	0.18800001	125	00:07:22	24	412	0.187	126	00:08:18	24
366	0.187	126	00:07:23	24	413	0.185	127	00:08:19	24
367	0.18800001	125	00:07:24	24	414	0.186	127	00:08:20	24
368	0.18800001	125	00:07:25	24	414	0	0	00:08:21	0
369	0.187	126	00:07:26	24	415	0.18800001	125	00:08:22	24
369	0	0	00:07:27	0	416	0.19800001	119	00:08:23	24
370	0.178	132	00:07:28	24	417	0.18800001	125	00:08:24	24
371	0.187	126	00:07:29	24	418	0.178	132	00:08:25	24
372	0.18800001	125	00:07:30	24	419	0.18800001	125	00:08:26	24
373	0.19700001	119	00:07:31	24	419	0	0	00:08:27	0
374	0.187	126	00:07:32	24	420	0.187	126	00:08:28	24
374	0	0	00:07:33	0	421	0.19700001	119	00:08:29	24
375	0.187	126	00:07:34	24	422	0.187	126	00:08:30	24
376	0.18800001	125	00:07:35	24	423	0.19700001	119	00:08:31	24
377	0.18800001	125	00:07:36	24	424	0.187	126	00:08:32	24
378	0.208	113.12	00:07:37	24	424	0	0	00:08:33	0
379	0.18800001	125	00:07:38	24	425	0.21800001	108	00:08:34	24
379	0	0	00:07:39	0	426	0.177	133	00:08:35	24
380	0.18800001	125	00:07:40	24	427	0.187	126	00:08:36	24
381	0.18800001	125	00:07:41	24	428	0.187	126	00:08:37	24
382	0.186	127	00:07:42	24	429	0.187	126	00:08:38	24
383	0.20700002	114	00:07:43	24	429	0	0	00:08:39	0
384	0.18800001	125	00:07:44	24	430	0.177	133	00:08:40	24
384	0	0	00:07:45	0	431	0.19600001	120	00:08:41	24
385	0.18900001	124	00:07:46	24	432	0.186	127	00:08:42	24
386	0.178	132	00:07:47	24	433	0.185	127	00:08:43	24
387	0.18800001	125	00:07:48	24	434	0.18900001	124	00:08:44	24
388	0.20700002	114	00:07:49	24	434	0	0	00:08:45	0
389	0.187	126	00:07:50	24	435	0.18800001	125	00:08:46	24
389	0	0	00:07:51	0	436	0.208	113.12	00:08:47	24
390	0.177	133	00:07:52	24	437	0.18800001	125	00:08:48	24
391	0.187	126	00:07:53	24	438	0.178	132	00:08:49	24
392	0.193	122	00:07:54	24	439	0.18800001	125	00:08:50	24
393	0.20700002	114	00:07:55	24	439	0	0	00:08:51	0
394	0.177	133	00:07:56	24	440	0.18800001	125	00:08:52	24
394	0	0	00:07:57	0	441	0.208	113.12	00:08:53	24
395	0.18800001	125	00:07:58	24	442	0.178	132	00:08:54	24
396	0.187	126	00:07:59	24	443	0.18800001	125	00:08:55	24
397	0.19700001	119	00:08:00	24	444	0.18800001	125	00:08:56	24
398	0.187	126	00:08:01	24	444	0	0	00:08:57	0
399	0.18800001	125	00:08:02	24	445	0.18800001	125	00:08:58	24
399	0	0	00:08:03	0	446	0.20700002	114	00:08:59	24
400	0.18800001	125	00:08:04	24	447	0.187	126	00:09:00	24
401	0.186	127	00:08:05	24	448	0.187	126	00:09:01	24
402	0.187	126	00:08:06	24	449	0.19700001	119	00:09:02	24
403	0.18800001	125	00:08:07	24	449	0	0	00:09:03	0
404	0.18800001	125	00:08:08	24	450	0.19600001	120	00:09:04	24
404	0	0	00:08:09	0	451	0.238	99	00:09:05	24
405	0.186	127	00:08:10	24	452	0.18800001	125	00:09:06	24
406	0.186	127	00:08:11	24	453	0.19600001	120	00:09:07	24
407	0.18800001	125	00:08:12	24	453	0	0	00:09:08	0
408	0.18800001	125	00:08:13	24	454	0.18	131	00:09:09	24
409	0.186	127	00:08:14	24	455	0.18800001	125	00:09:10	24
409	0	0	00:08:15	0	456	0.19700001	119	00:09:11	24
410	0.19600001	120	00:08:16	24	457	0.18800001	125	00:09:12	24
411	0.187	126	00:08:17	24	458	0.178	132	00:09:13	24
458	0	0	00:09:14	0					
459	0.18800001	125	00:09:15	24					
460	0.187	126	00:09:16	24					
461	0.19600001	120	00:09:17	24					
462	0.186	127	00:09:18	24					
463	0.19600001	120	00:09:19	24					
463	0	0	00:09:20	0					
464	0.186	127	00:09:21	24					
465	0.186	127	00:09:22	24					
466	0.187	126	00:09:23	24					
467	0.18800001	125	00:09:24	24					
468	0.18800001	125	00:09:25	24					
468	0	0	00:09:26	0					
469	0.186	127	00:09:27	24					
470	0.22800002	103	00:09:28	24					
471	0.18800001	125	00:09:29	24					
472	0.18800001	125	00:09:30	24					
473	0.186	127	00:09:31	24					
473	0	0	00:09:32	0					
474	0.187	126	00:09:33	24					
475	0.187	126	00:09:34	24					
476	0.19700001	119	00:09:35	24					
477	0.187	126	00:09:36	24					
478	0.177	133	00:09:37	24					
478	0	0	00:09:38	0					
479	0.187	126	00:09:39	24					
480	0.187	126	00:09:40	24					
481	0.19700001	119	00:09:41	24					
482	0.187	126	00:09:42	24					
483	0.19700001	119	00:09:43	24					
483	0	0	00:09:44	0					
484	0.187	126	00:09:45	24					
485	0.20700002	114	00:09:46	24					
486	0.18800001	125	00:09:47	24					
487	0.18800001	125	00:09:48	24					
488	0.18800001	125	00:09:49	24					
488	0	0	00:09:50	0					
489	0.18800001	125	00:09:51	24					
490	0.178	132	00:09:52	24					
491	0.19700001	119	00:09:53	24					
492	0.18800001	125	00:09:54	24					
493	0.18800001	125	00:09:55	24					
493	0	0	00:09:56	0					
494	0.187	126	00:09:57	24					
495	0.187	126	00:09:58	24					
496	0.19900002	118	00:09:59	24					
497	0.187	126	00:10:00	24					
498	0.177	133	00:10:01	24					

Sum	94.905003	61654.3506
Average	0.1905723	123.803917

Measurement
Webserver

compressed 1:3.6091 = 72%

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.11000001	134	00:00:04	15
2	0.10900001	135	00:00:05	15
3	0.116	126.94	00:00:06	15
4	0.119	124	00:00:07	15
5	0.119	124	00:00:08	15
6	0.11000001	134	00:00:09	15
7	0.11000001	134	00:00:10	15
7	0	0	00:00:11	0
8	0.112	131	00:00:12	15
9	0.119	124	00:00:13	15
10	0.10900001	135	00:00:14	15
11	0.11100001	133	00:00:15	15
12	0.12900001	114	00:00:16	15
13	0.11000001	134	00:00:17	15
14	0.10900001	135	00:00:18	15
15	0.10900001	135	00:00:19	15
15	0	0	00:00:20	0
16	0.119	124	00:00:21	15
17	0.10900001	135	00:00:22	15
18	0.119	124	00:00:23	15
19	0.119	124	00:00:24	15
20	0.11800001	125	00:00:25	15
21	0.11800001	125	00:00:26	15
22	0.11800001	125	00:00:27	15
23	0.119	124	00:00:28	15
23	0	0	00:00:29	0
24	0.119	124	00:00:30	15
25	0.119	124	00:00:31	15
26	0.11800001	125	00:00:32	15
27	0.11800001	125	00:00:33	15
28	0.11800001	125	00:00:34	15
29	0.119	124	00:00:35	15
30	0.119	124	00:00:36	15
30	0	0	00:00:37	0
31	0.119	124	00:00:38	15
32	0.11800001	125	00:00:39	15
33	0.11800001	125	00:00:40	15
34	0.11800001	125	00:00:41	15
35	0.15900001	92.61	00:00:42	15
36	0.11000001	134	00:00:43	15
37	0.11000001	134	00:00:44	15
38	0.119	124	00:00:45	15
38	0	0	00:00:46	0
39	0.13900001	106	00:00:47	15
40	0.119	124	00:00:48	15
41	0.10900001	135	00:00:49	15
42	0.11000001	134	00:00:50	15
43	0.12000001	123	00:00:51	15
44	0.10900001	135	00:00:52	15
45	0.11800001	125	00:00:53	15
45	0	0	00:00:54	0
46	0.11800001	125	00:00:55	15
47	0.11800001	125	00:00:56	15
48	0.13900001	106	00:00:57	15
49	0.119	124	00:00:58	15
50	0.11800001	125	00:00:59	15
51	0.11800001	125	00:01:00	15
52	0.11800001	125	00:01:01	15
53	0.11800001	125	00:01:02	15
53	0	0	00:01:03	0
54	0.119	124	00:01:04	15
55	0.119	124	00:01:05	15
56	0.119	124	00:01:06	15
57	0.11800001	125	00:01:07	15
58	0.11800001	125	00:01:08	15
59	0.149	99	00:01:09	15
60	0.119	124	00:01:10	15
60	0	0	00:01:11	0
61	0.119	124	00:01:12	15
62	0.11800001	125	00:01:13	15
63	0.11800001	125	00:01:14	15
64	0.11800001	125	00:01:15	15
65	0.11800001	125	00:01:16	15
66	0.119	124	00:01:17	15
67	0.119	124	00:01:18	15
68	0.119	124	00:01:19	15
68	0	0	00:01:20	0
69	0.11800001	125	00:01:21	15
70	0.11800001	125	00:01:22	15
71	0.11800001	125	00:01:23	15
72	0.119	124	00:01:24	15
73	0.13900001	106	00:01:25	15
74	0.119	124	00:01:26	15
75	0.11800001	125	00:01:27	15
75	0	0	00:01:28	0
76	0.123	120	00:01:29	15
77	0.11800001	125	00:01:30	15
78	0.11800001	125	00:01:31	15
79	0.119	124	00:01:32	15
80	0.119	124	00:01:33	15
81	0.11800001	125	00:01:34	15
82	0.11800001	125	00:01:35	15
83	0.11800001	125	00:01:36	15
83	0	0	00:01:37	0
84	0.11800001	125	00:01:38	15
85	0.119	124	00:01:39	15
86	0.119	124	00:01:40	15
87	0.119	124	00:01:41	15
88	0.11800001	125	00:01:42	15
89	0.11800001	125	00:01:43	15
90	0.11800001	125	00:01:44	15
91	0.11800001	125	00:01:45	15
91	0	0	00:01:46	0
92	0.119	124	00:01:47	15
93	0.119	124	00:01:48	15
94	0.11800001	125	00:01:49	15
95	0.11800001	125	00:01:50	15
96	0.11800001	125	00:01:51	15
97	0.11800001	125	00:01:52	15
98	0.119	124	00:01:53	15
98	0	0	00:01:54	0
99	0.12900001	114	00:01:55	15
100	0.17400001	85	00:01:56	15
101	0.10900001	135	00:01:57	15
102	0.10900001	135	00:01:58	15
103	0.119	124	00:01:59	15
104	0.10900001	135	00:02:00	15
105	0.11000001	134	00:02:01	15
106	0.11000001	134	00:02:02	15
106	0	0	00:02:03	0
107	0.11700001	126	00:02:04	15
108	0.119	124	00:02:05	15
109	0.10900001	135	00:02:06	15
110	0.12000001	123	00:02:07	15
111	0.11000001	134	00:02:08	15
112	0.119	124	00:02:09	15
113	0.119	124	00:02:10	15
114	0.10900001	135	00:02:11	15
114	0	0	00:02:12	0
115	0.13800001	107	00:02:13	15
116	0.11000001	134	00:02:14	15
117	0.11000001	134	00:02:15	15
118	0.10900001	135	00:02:16	15
119	0.10900001	135	00:02:17	15
120	0.10900001	135	00:02:18	15
121	0.10900001	135	00:02:19	15
122	0.12000001	123	00:02:20	15
122	0	0	00:02:21	0
123	0.119	124	00:02:22	15
124	0.11800001	125	00:02:23	15
125	0.11800001	125	00:02:24	15
126	0.11800001	125	00:02:25	15
127	0.11800001	125	00:02:26	15
128	0.148	99	00:02:27	15
129	0.119	124	00:02:28	15
129	0	0	00:02:29	0
130	0.11800001	125	00:02:30	15
131	0.11800001	125	00:02:31	15
132	0.11800001	125	00:02:32	15
133	0.11800001	125	00:02:33	15
134	0.119	124	00:02:34	15
135	0.13900001	106	00:02:35	15
136	0.119	124	00:02:36	15
137	0.11800001	125	00:02:37	15
137	0	0	00:02:38	0
138	0.11800001	125	00:02:39	15
139	0.11800001	125	00:02:40	15
140	0.119	124	00:02:41	15
141	0.119	124	00:02:42	15
142	0.119	124	00:02:43	15
143	0.11800001	125	00:02:44	15
144	0.126	117	00:02:45	15
144	0	0	00:02:46	0
145	0.10900001	135	00:02:47	15
146	0.10900001	135	00:02:48	15
147	0.11000001	134	00:02:49	15
148	0.11000001	134	00:02:50	15
149	0.119	124	00:02:51	15
150	0.11800001	125	00:02:52	15
151	0.11800001	125	00:02:53	15
152	0.11800001	125	00:02:54	15
152	0	0	00:02:55	0
153	0.119	124	00:02:56	15
154	0.119	124	00:02:57	15
155	0.11800001	125	00:02:58	15
156	0.11800001	125	00:02:59	15
157	0.11800001	125	00:03:00	15
158	0.123	120	00:03:01	15
159	0.128	115	00:03:02	15
160	0.11800001	125	00:03:03	15
160	0	0	00:03:04	0
161	0.11800001	125	00:03:05	15
162	0.119	124	00:03:06	15
163	0.119	124	00:03:07	15
164	0.119	124	00:03:08	15
165	0.11800001	125	00:03:09	15
166	0.11800001	125	00:03:10	15
167	0.11800001	125	00:03:11	15
167	0	0	00:03:12	0
168	0.11800001	125	00:03:13	15
169	0.119	124	00:03:14	15
170	0.119	124	00:03:15	15
171	0.11800001	125	00:03:16	15
172	0.11800001	125	00:03:17	15
173	0.11800001	125	00:03:18	15
174	0.11800001	125	00:03:19	15
175	0.119	124	00:03:20	15
175	0	0	00:03:21	0
176	0.11800001	125	00:03:22	15
177	0.11800001	125	00:03:23	15
178	0.13800001	107	00:03:24	15
179	0.11800001	125	00:03:25	15
180	0.11800001	125	00:03:26	15
181	0.11800001	125	00:03:27	15
182	0.119	124	00:03:28	15
182	0	0	00:03:29	0
183	0.119	124	00:03:30	15
184	0.11800001	125	00:03:31	15
185	0.11800001	125	00:03:32	15
186	0.11800001	125	00:03:33	15
187	0.11800001	125	00:03:34	15
188	0.119	124	00:03:35	15
189	0.119	124	00:03:36	15
190	0.11800001	125	00:03:37	15
190	0	0	00:03:38	0
191	0.11800001	125	00:03:39	15
192	0.11800001	125	00:03:40	15
193	0.11800001	125	00:03:41	15
194	0.119	124	00:03:42	15
195	0.119	124	00:03:43	15
196	0.119	124	00:03:44	15
197	0.11800001	125	00:03:45	15
198	0.11800001	125	00:03:46	15
198	0	0	00:03:47	0
199	0.11800001	125	00:03:48	15
200	0.11800001	125	00:03:49	15
201	0.119	124	00:03:50	15
202	0.119	124	00:03:51	15
203	0.11800001	125	00:03:52	15
204	0.11800001	125	00:03:53	15
205	0.13800001	107	00:03:54	15
205	0	0	00:03:55	0
206	0.128	115	00:03:56	15
207	0.12900001	114	00:03:57	15
208	0.11000001	134	00:03:58	15
209	0.10900001	135	00:03:59	15
210	0.10900001	135	00:04:00	15
211	0.10900001	135	00:04:01	15
212	0.10900001	135	00:04:02	15
213	0.12000001	123	00:04:03	15
213	0	0	00:04:04	0
214	0.11000001	134	00:04:05	15
215	0.119	124	00:04:06	15
216	0.10900001	135	00:04:07	15
217	0.10900001	135	00:04:08	15
218	0.11000001	134	00:04:09	15
219	0.11000001	134	00:04:10	15
220	0.119	124	00:04:11	15
221	0.11800001	125	00:04:12	15
221	0	0	00:04:13	0
222	0.11800001	125	00:04:14	15
223	0.11800001	125	00:04:15	15
224	0.119	124	00:04:16	15
225	0.119	124	00:04:17	15
226	0.119	124	00:04:18	15
227	0.11800001	125	00:04:19	15
228	0.11800001	125	00:04:20	15
228	0	0	00:04:21	0
229	0.11800001	125	00:04:22	15
230	0.11800001	125	00:04:23	15
231	0.119	124	00:04:24	15
232	0.13900001	106	00:04:25	15
233	0.11800001	125	00:04:26	15
234	0.11800001	125	00:04:27	15
235	0.126	117	00:04:28	15
236	0.128	115	00:04:29	15
236	0	0	00:04:30	0
237	0.11800001	125	00:04:31	15
238	0.11000001	134	00:04:32	15
239	0.11000001	134	00:04:33	15
240	0.119	124	00:04:34	15
241	0.10900001	135	00:04:35	15
242	0.10900001	135	00:04:36	15
243	0.119	124	00:04:37	15
244	0.119	124	00:04:38	15
244	0	0	00:04:39	0
245	0.119	124	00:04:40	15
246	0.11800001	125	00:04:41	15
247	0.11800001	125	00:04:42	15
248	0.11800001	125	00:04:43	15
249	0.119	124	00:04:44	15
250	0.119	124	00:04:45	15
251	0.119	124	00:04:46	15
251	0	0	00:04:47	0
252	0.11800001	125	00:04:48	15
253	0.10900001	135	00:04:49	15
254	0.10900001	135	00:04:50	15
255	0.11000001	134	00:04:51	15
256	0.11000001	134	00:04:52	15
257	0.10900001	135	00:04:53	15
258	0.11800001	125	00:04:54	15
259	0.11800001	125	00:04:55	15
259	0	0	00:04:56	0
260	0.11800001	125	00:04:57	15
261	0.11800001	125	00:04:58	15
262	0.119	124	00:04:59	15
263	0.119	124	00:05:00	15
264	0.13000001	113	00:05:01	15
265	0.119	124	00:05:02	15
265	0	0	00:05:03	0
266	0.192	77	00:05:04	15
267	0.10900001	135	00:05:05	15
268	0.116	126.94	00:05:06	15
269	0.149	99	00:05:07	15
270	0.10900001	135	00:05:08	15
271	0.12000001	123	00:05:09	15
272	0.11000001	134	00:05:10	15
272	0	0	00:05:11	0
273	0.10900001	135	00:05:12	15
274	0.10900001	135	00:05:13	15
275	0.10900001	135	00:05:14	15
276	0.11800001	125	00:05:15	15
277	0.13000001	113	00:05:16	15
278	0.11800001	125	00:05:17	15
279	0.11800001	125	00:05:18	15
280	0.11800001	125	00:05:19	15
280	0	0	00:05:20	0
281	0.11800001	125	00:05:21	15
282	0.119	124	00:05:22	15
283	0.119	124	00:05:23	15
284	0.119	124	00:05:24	15
285	0.11800001	125	00:05:25	15
286	0.11800001	125	00:05:26	15
287	0.11800001	125	00:05:27	15
287	0	0	00:05:28	0
288	0.11800001	125	00:05:29	15
289	0.119	124	00:05:30	15
290	0.12900001	114	00:05:31	15
291	0.11800001	125	00:05:32	15
292	0.15900001	92.61	00:05:33	15
293	0.10900001	135	00:05:34	15
294	0.10900001	135	00:05:35	15
295	0.11000001	134	00:05:36	15
295	0	0	00:05:37	0
296	0.14	105	00:05:38	15
297	0.10900001	135	00:05:39	15
298	0.10900001	135	00:05:40	15
299	0.11800001	125	00:05:41	15
300	0.119	124	00:05:42	15
301	0.125	117.8	00:05:43	15
302	0.127	116	00:05:44	15
303	0.12000001	123	00:05:45	15
303	0	0	00:05:46	0
304	0.10900001	135	00:05:47	15
305	0.119	124	00:05:48	15
306	0.10900001	135	00:05:49	15
307	0.11000001	134	00:05:50	15
308	0.11000001	134	00:05:51	15
309	0.11800001	125	00:05:52	15
310	0.11800001	125	00:05:53	15
311	0.11800001	125	00:05:54	15
311	0	0	00:05:55	0
312	0.126	117	00:05:56	15
313	0.119	124	00:05:57	15
314	0.119	124	00:05:58	15
315	0.11800001	125	00:05:59	15
316	0.11800001	125	00:06:00	15
317	0.128	115	00:06:01	15
317	0	0	00:06:02	0
318	0.126	117	00:06:03	15
319	0.119	124	00:06:04	15
320	0.11800001	125	00:06:05	15
321	0.11800001	125	00:06:06	15
322	0.11800001	125	00:06:07	15
323	0.11800001	125	00:06:08	15
324	0.119	124	00:06:09	15
325	0.119	124	00:06:10	15
325	0	0	00:06:11	0
326	0.11800001	125	00:06:12	15
327	0.11800001	125	00:06:13	15
328	0.11700001	126	00:06:14	15
329	0.11800001	125	00:06:15	15
330	0.119	124	00:06:16	15
331	0.119	124	00:06:17	15
332	0.11800001	125	00:06:18	15
333	0.11800001	125	00:06:19	15
333	0	0	00:06:20	0
334	0.11800001	125	00:06:21	15
335	0.11800001	125	00:06:22	15
336	0.119	124	00:06:23	15
337	0.119	124	00:06:24	15
338	0.13900001	106	00:06:25	15
339	0.11800001	125	00:06:26	15
340	0.11800001	125	00:06:27	15
340	0	0	00:06:28	0
341	0.126	117	00:06:29	15
342	0.11800001	125	00:06:30	15
343	0.119	124	00:06:31	15
344	0.119	124	00:06:32	15
345	0.11800001	125	00:06:33	15
346	0.11800001	125	00:06:34	15
347	0.11800001	125	00:06:35	15
348	0.11800001	125	00:06:36	15
348	0	0	00:06:37	0
349	0.119	124	00:06:38	15
350	0.119	124	00:06:39	15
351	0.11800001	125	00:06:40	15
352	0.11800001	125	00:06:41	15
353	0.11800001	125	00:06:42	15
354	0.11800001	125	00:06:43	15
355	0.11800001	125	00:06:44	15
355	0	0	00:06:45	0
356	0.119	124	00:06:46	15
357	0.119	124	00:06:47	15
358	0.128	115	00:06:48	15
359	0.11800001	125	00:06:49	15
360	0.11800001	125	00:06:50	15
361	0.13800001	107	00:06:51	15
362	0.125	117.8	00:06:52	15
363	0.11000001	134	00:06:53	15
363	0	0	00:06:54	0
364	0.11700001	126	00:06:55	15
365	0.147	100	00:06:56	15
366	0.11800001	125	00:06:57	15
367	0.10900001	135	00:06:58	15
368	0.12000001	123	00:06:59	15
369	0.11000001	134	00:07:00	15
369	0	0	00:07:01	0
370	0.12100001	122	00:07:02	15
371	0.119	124	00:07:03	15
372	0.11800001	125	00:07:04	15
373	0.11800001	125	00:07:05	15
374	0.11800001	125	00:07:06	15
375	0.119	124	00:07:07	15
376	0.119	124	00:07:08	15
377	0.119	124	00:07:09	15
377	0	0	00:07:10	0
378	0.11800001	125	00:07:11	15
379	0.11800001	125	00:07:12	15
380	0.11800001	125	00:07:13	15
381	0.11800001	125	00:07:14	15
382	0.119	124	00:07:15	15
383	0.119	124	00:07:16	15
384	0.11800001	125	00:07:17	15
385	0.11800001	125	00:07:18	15
385	0	0	00:07:19	0
386	0.11800001	125	00:07:20	15
387	0.11800001	125	00:07:21	15

388	0.119	124	00:07:22	15	437	0	0	00:08:18	0
389	0.119	124	00:07:23	15	438	0.10900001	135	00:08:19	15
390	0.119	124	00:07:24	15	439	0.10900001	135	00:08:20	15
391	0.13800001	107	00:07:25	15	440	0.11000001	134	00:08:21	15
392	0.11800001	125	00:07:26	15	441	0.10900001	135	00:08:22	15
392	0	0	00:07:27	0	442	0.10900001	135	00:08:23	15
393	0.11800001	125	00:07:28	15	443	0.11800001	125	00:08:24	15
394	0.119	124	00:07:29	15	444	0.11800001	125	00:08:25	15
395	0.119	124	00:07:30	15	445	0.119	124	00:08:26	15
396	0.119	124	00:07:31	15	445	0	0	00:08:27	0
397	0.11800001	125	00:07:32	15	446	0.119	124	00:08:28	15
398	0.11800001	125	00:07:33	15	447	0.119	124	00:08:29	15
399	0.12900001	114	00:07:34	15	448	0.11800001	125	00:08:30	15
400	0.119	124	00:07:35	15	449	0.11800001	125	00:08:31	15
400	0	0	00:07:36	0	450	0.11800001	125	00:08:32	15
401	0.119	124	00:07:37	15	451	0.119	124	00:08:33	15
402	0.119	124	00:07:38	15	452	0.119	124	00:08:34	15
403	0.11800001	125	00:07:39	15	452	0	0	00:08:35	0
404	0.11800001	125	00:07:40	15	453	0.119	124	00:08:36	15
405	0.11800001	125	00:07:41	15	454	0.11800001	125	00:08:37	15
406	0.125	117.8	00:07:42	15	455	0.11800001	125	00:08:38	15
407	0.11800001	125	00:07:43	15	456	0.15900001	92.61	00:08:39	15
407	0	0	00:07:44	0	457	0.10900001	135	00:08:40	15
408	0.119	124	00:07:45	15	458	0.11000001	134	00:08:41	15
409	0.119	124	00:07:46	15	459	0.11000001	134	00:08:42	15
410	0.11800001	125	00:07:47	15	460	0.13900001	106	00:08:43	15
411	0.11800001	125	00:07:48	15	460	0	0	00:08:44	0
412	0.11800001	125	00:07:49	15	461	0.10900001	135	00:08:45	15
413	0.11800001	125	00:07:50	15	462	0.11100001	133	00:08:46	15
414	0.119	124	00:07:51	15	463	0.115	128	00:08:47	15
415	0.119	124	00:07:52	15	464	0.12900001	114	00:08:48	15
415	0	0	00:07:53	0	465	0.11000001	134	00:08:49	15
416	0.119	124	00:07:54	15	466	0.127	116	00:08:50	15
417	0.11800001	125	00:07:55	15	467	0.125	117.8	00:08:51	15
418	0.11800001	125	00:07:56	15	467	0	0	00:08:52	0
419	0.11800001	125	00:07:57	15	468	0.10900001	135	00:08:53	15
420	0.11800001	125	00:07:58	15	469	0.10900001	135	00:08:54	15
421	0.119	124	00:07:59	15	470	0.11000001	134	00:08:55	15
422	0.13900001	106	00:08:00	15	471	0.12000001	123	00:08:56	15
422	0	0	00:08:01	0	472	0.10900001	135	00:08:57	15
423	0.123	120	00:08:02	15	473	0.119	124	00:08:58	15
424	0.127	116	00:08:03	15	474	0.10900001	135	00:08:59	15
425	0.11800001	125	00:08:04	15	475	0.13900001	106	00:09:00	15
426	0.119	124	00:08:05	15	475	0	0	00:09:01	0
427	0.119	124	00:08:06	15	476	0.13700001	107	00:09:02	15
428	0.11800001	125	00:08:07	15	477	0.10900001	135	00:09:03	15
429	0.11800001	125	00:08:08	15	478	0.12000001	123	00:09:04	15
429	0	0	00:08:09	0	479	0.11000001	134	00:09:05	15
430	0.11800001	125	00:08:10	15	480	0.119	124	00:09:06	15
431	0.11800001	125	00:08:11	15	481	0.10900001	135	00:09:07	15
432	0.119	124	00:08:12	15	481	0	0	00:09:08	0
433	0.119	124	00:08:13	15	482	0.10900001	135	00:09:09	15
434	0.119	124	00:08:14	15	483	0.10900001	135	00:09:10	15
435	0.11800001	125	00:08:15	15	484	0.11000001	134	00:09:11	15
436	0.126	117	00:08:16	15	485	0.112	131	00:09:12	15
437	0.11800001	125	00:08:17	15	486	0.108	136	00:09:13	15
487	0.10900001	135	00:09:14	15					
488	0.10900001	135	00:09:15	15					
489	0.119	124	00:09:16	15					
489	0	0	00:09:17	0					
490	0.13900001	106	00:09:18	15					
491	0.11000001	134	00:09:19	15					
492	0.10900001	135	00:09:20	15					
493	0.112	131	00:09:21	15					
494	0.10900001	135	00:09:22	15					
495	0.11000001	134	00:09:23	15					
496	0.10900001	135	00:09:24	15					
497	0.119	124	00:09:25	15					
497	0	0	00:09:26	0					
498	0.11800001	125	00:09:27	15					
499	0.11800001	125	00:09:28	15					
500	0.119	124	00:09:29	15					
501	0.119	124	00:09:30	15					
502	0.119	124	00:09:31	15					
503	0.11800001	125	00:09:32	15					
504	0.11800001	125	00:09:33	15					
505	0.11800001	125	00:09:34	15					
505	0	0	00:09:35	0					
506	0.11800001	125	00:09:36	15					
507	0.119	124	00:09:37	15					
508	0.119	124	00:09:38	15					
509	0.119	124	00:09:39	15					
510	0.11800001	125	00:09:40	15					
511	0.11800001	125	00:09:41	15					
512	0.11800001	125	00:09:42	15					
512	0	0	00:09:43	0					
513	0.119	124	00:09:44	15					
514	0.119	124	00:09:45	15					
515	0.11800001	125	00:09:46	15					
516	0.11800001	125	00:09:47	15					
517	0.149	99	00:09:48	15					
518	0.11800001	125	00:09:49	15					
519	0.11800001	125	00:09:50	15					
520	0.119	124	00:09:51	15					
520	0	0	00:09:52	0					
521	0.119	124	00:09:53	15					
522	0.11800001	125	00:09:54	15					
523	0.11800001	125	00:09:55	15					
524	0.11800001	125	00:09:56	15					
525	0.11800001	125	00:09:57	15					
526	0.119	124	00:09:58	15					
527	0.119	124	00:09:59	15					
527	0	0	00:10:00	0					
528	0.119	124	00:10:01	15					
529	0.049	242	00:10:02	12					

		66083.5102
Sum	62.6480031	3
		124.921569
Average	0.11842723	4

Measurement Web Server

compressed 1:32.34365 = 97%

Total Repeats	Average Duration (seconds)	Average Speed (kBytes/sec)	Time (HH:MM:SS)	Average Size (kBytes)
0	0	0	00:00:01	0
0	0	0	00:00:02	0
0	0	0	00:00:03	0
1	0.021	100.095	00:00:04	2.102
2	0.022	95.545006	00:00:05	2.102
3	0.021	100.095	00:00:06	2.102
4	0.022	95.545006	00:00:07	2.102
5	0.021	100.095	00:00:08	2.102
6	0.074	28.405	00:00:09	2.102
7	0.023	91.39101	00:00:10	2.102
8	0.021	100.095	00:00:11	2.102
9	0.022	95.545006	00:00:12	2.102
10	0.023	91.39101	00:00:13	2.102
11	0.07300001	28.795002	00:00:14	2.102
12	0.033	63.697002	00:00:15	2.102
13	0.021	100.095	00:00:16	2.102
14	0.022	95.545006	00:00:17	2.102
15	0.062	33.903	00:00:18	2.102
16	0.021	100.095	00:00:19	2.102
17	0.022	95.545006	00:00:20	2.102
18	0.021	100.095	00:00:21	2.102
19	0.022	95.545006	00:00:22	2.102
20	0.021	100.095	00:00:23	2.102
21	0.021	100.095	00:00:24	2.102
21	0	0	00:00:25	0
22	0.021	100.095	00:00:26	2.102
23	0.038	55.316	00:00:27	2.102
24	0.014	150.143	00:00:28	2.102
25	0.021	100.095	00:00:29	2.102
26	0.021	100.095	00:00:30	2.102
27	0.021	100.095	00:00:31	2.102
28	0.02	105.100006	00:00:32	2.102
29	0.021	100.095	00:00:33	2.102
30	0.021	100.095	00:00:34	2.102
31	0.021	100.095	00:00:35	2.102
32	0.019	110.632	00:00:36	2.102
33	0.021	100.095	00:00:37	2.102
34	0.022	95.545006	00:00:38	2.102
35	0.021	100.095	00:00:39	2.102
36	0.022	95.545006	00:00:40	2.102
37	0.021	100.095	00:00:41	2.102
38	0.028	75.07101	00:00:42	2.102
39	0.05	42.04	00:00:43	2.102
40	0.022	95.545006	00:00:44	2.102
41	0.021	100.095	00:00:45	2.102
42	0.022	95.545006	00:00:46	2.102
43	0.021	100.095	00:00:47	2.102
44	0.012	175.167	00:00:48	2.102
45	0.021	100.095	00:00:49	2.102
46	0.022	95.545006	00:00:50	2.102
47	0.021	100.095	00:00:51	2.102
48	0.021	100.095	00:00:52	2.102
49	0.021	100.095	00:00:53	2.102
49	0	0	00:00:54	0
50	0.021	100.095	00:00:55	2.102
51	0.021	100.095	00:00:56	2.102
52	0.011	191.091	00:00:57	2.102
53	0.021	100.095	00:00:58	2.102
54	0.021	100.095	00:00:59	2.102
55	0.022	95.545006	00:01:00	2.102
56	0.021	100.095	00:01:01	2.102
57	0.021	100.095	00:01:02	2.102
58	0.037	56.811	00:01:03	2.102
59	0.041	51.268	00:01:04	2.102
60	0.012	175.167	00:01:05	2.102
61	0.022	95.545006	00:01:06	2.102
62	0.021	100.095	00:01:07	2.102
63	0.021	100.095	00:01:08	2.102
64	0.021	100.095	00:01:09	2.102
65	0.022	95.545006	00:01:10	2.102
66	0.021	100.095	00:01:11	2.102
67	0.021	100.095	00:01:12	2.102
68	0.021	100.095	00:01:13	2.102
69	0.022	95.545006	00:01:14	2.102
70	0.011	191.091	00:01:15	2.102
71	0.022	95.545006	00:01:16	2.102
72	0.012	175.167	00:01:17	2.102
73	0.022	95.545006	00:01:18	2.102
74	0.072	29.194002	00:01:19	2.102
75	0.021	100.095	00:01:20	2.102
76	0.011	191.091	00:01:21	2.102
77	0.021	100.095	00:01:22	2.102
78	0.021	100.095	00:01:23	2.102
78	0	0	00:01:24	0
79	0.021	100.095	00:01:25	2.102
80	0.021	100.095	00:01:26	2.102
81	0.022	95.545006	00:01:27	2.102
82	0.021	100.095	00:01:28	2.102
83	0.021	100.095	00:01:29	2.102
84	0.012	175.167	00:01:30	2.102
85	0.021	100.095	00:01:31	2.102
86	0.021	100.095	00:01:32	2.102
87	0.021	100.095	00:01:33	2.102
88	0.021	100.095	00:01:34	2.102
89	0.061	34.459003	00:01:35	2.102
90	0.025	84.08	00:01:36	2.102
91	0.024	87.58301	00:01:37	2.102
92	0.022	95.545006	00:01:38	2.102
93	0.021	100.095	00:01:39	2.102
94	0.022	95.545006	00:01:40	2.102
95	0.021	100.095	00:01:41	2.102
96	0.022	95.545006	00:01:42	2.102
97	0.021	100.095	00:01:43	2.102
98	0.022	95.545006	00:01:44	2.102
99	0.021	100.095	00:01:45	2.102
100	0.021	100.095	00:01:46	2.102
101	0.021	100.095	00:01:47	2.102
102	0.021	100.095	00:01:48	2.102
103	0.039	53.897003	00:01:49	2.102
104	0.041	51.268	00:01:50	2.102
105	0.021	100.095	00:01:51	2.102
106	0.021	100.095	00:01:52	2.102
106	0	0	00:01:53	0
107	0.022	95.545006	00:01:54	2.102
108	0.021	100.095	00:01:55	2.102
109	0.021	100.095	00:01:56	2.102
110	0.021	100.095	00:01:57	2.102
111	0.022	95.545006	00:01:58	2.102
112	0.021	100.095	00:01:59	2.102
113	0.021	100.095	00:02:00	2.102
114	0.021	100.095	00:02:01	2.102
115	0.024	87.58301	00:02:02	2.102
116	0.026	80.846	00:02:03	2.102
117	0.021	100.095	00:02:04	2.102
118	0.037	56.811	00:02:05	2.102
119	0.015	140.13301	00:02:06	2.102
120	0.021	100.095	00:02:07	2.102
121	0.021	100.095	00:02:08	2.102
122	0.021	100.095	00:02:09	2.102
123	0.022	95.545006	00:02:10	2.102
124	0.011	191.091	00:02:11	2.102
125	0.022	95.545006	00:02:12	2.102
126	0.021	100.095	00:02:13	2.102
127	0.022	95.545006	00:02:14	2.102
128	0.011	191.091	00:02:15	2.102
129	0.022	95.545006	00:02:16	2.102
130	0.021	100.095	00:02:17	2.102
131	0.022	95.545006	00:02:18	2.102
132	0.02	105.100006	00:02:19	2.102
133	0.037	56.811	00:02:20	2.102
134	0.041	51.268	00:02:21	2.102
134	0	0	00:02:22	0
135	0.018	116.77801	00:02:23	2.102
136	0.021	100.095	00:02:24	2.102
137	0.021	100.095	00:02:25	2.102
138	0.021	100.095	00:02:26	2.102
139	0.023	91.39101	00:02:27	2.102
140	0.011	191.091	00:02:28	2.102
141	0.021	100.095	00:02:29	2.102
142	0.021	100.095	00:02:30	2.102
143	0.021	100.095	00:02:31	2.102
144	0.021	100.095	00:02:32	2.102
145	0.021	100.095	00:02:33	2.102
146	0.022	95.545006	00:02:34	2.102
147	0.021	100.095	00:02:35	2.102
148	0.012	175.167	00:02:36	2.102
149	0.035	60.057003	00:02:37	2.102
150	0.041	51.268	00:02:38	2.102
151	0.021	100.095	00:02:39	2.102
152	0.021	100.095	00:02:40	2.102
153	0.021	100.095	00:02:41	2.102
154	0.021	100.095	00:02:42	2.102
155	0.021	100.095	00:02:43	2.102
156	0.011	191.091	00:02:44	2.102
157	0.021	100.095	00:02:45	2.102
158	0.021	100.095	00:02:46	2.102
159	0.021	100.095	00:02:47	2.102
160	0.021	100.095	00:02:48	2.102
161	0.021	100.095	00:02:49	2.102
162	0.021	100.095	00:02:50	2.102
163	0.023	91.39101	00:02:51	2.102
164	0.027	77.852005	00:02:52	2.102
164	0	0	00:02:53	0
165	0.013	161.692	00:02:54	2.102
166	0.021	100.095	00:02:55	2.102
167	0.022	95.545006	00:02:56	2.102
168	0.021	100.095	00:02:57	2.102
169	0.021	100.095	00:02:58	2.102
170	0.021	100.095	00:02:59	2.102
171	0.022	95.545006	00:03:00	2.102
172	0.011	191.091	00:03:01	2.102
173	0.022	95.545006	00:03:02	2.102
174	0.021	100.095	00:03:03	2.102
175	0.022	95.545006	00:03:04	2.102
176	0.021	100.095	00:03:05	2.102
177	0.022	95.545006	00:03:06	2.102
178	0.021	100.095	00:03:07	2.102
179	0.037	56.811	00:03:08	2.102
180	0.031	67.806	00:03:09	2.102
181	0.021	100.095	00:03:10	2.102
182	0.022	95.545006	00:03:11	2.102
183	0.021	100.095	00:03:12	2.102
184	0.021	100.095	00:03:13	2.102
185	0.021	100.095	00:03:14	2.102
186	0.021	100.095	00:03:15	2.102
187	0.021	100.095	00:03:16	2.102
188	0.011	191.091	00:03:17	2.102
189	0.021	100.095	00:03:18	2.102
190	0.021	100.095	00:03:19	2.102
191	0.021	100.095	00:03:20	2.102
192	0.021	100.095	00:03:21	2.102
193	0.021	100.095	00:03:22	2.102
193	0	0	00:03:23	0
194	0.034	61.824	00:03:24	2.102
195	0.041	51.268	00:03:25	2.102
196	0.012	175.167	00:03:26	2.102
197	0.021	100.095	00:03:27	2.102
198	0.022	95.545006	00:03:28	2.102
199	0.021	100.095	00:03:29	2.102
200	0.022	95.545006	00:03:30	2.102
201	0.021	100.095	00:03:31	2.102
202	0.021	100.095	00:03:32	2.102
203	0.021	100.095	00:03:33	2.102
204	0.012	175.167	00:03:34	2.102
205	0.021	100.095	00:03:35	2.102
206	0.022	95.545006	00:03:36	2.102
207	0.021	100.095	00:03:37	2.102
208	0.022	95.545006	00:03:38	2.102
209	0.021	100.095	00:03:39	2.102
210	0.061	34.459003	00:03:40	2.102
211	0.021	100.095	00:03:41	2.102
212	0.021	100.095	00:03:42	2.102
213	0.021	100.095	00:03:43	2.102
214	0.021	100.095	00:03:44	2.102
215	0.026	80.846	00:03:45	2.102
216	0.021	100.095	00:03:46	2.102
217	0.022	95.545006	00:03:47	2.102
218	0.021	100.095	00:03:48	2.102
219	0.021	100.095	00:03:49	2.102
220	0.011	191.091	00:03:50	2.102
221	0.021	100.095	00:03:51	2.102
222	0.021	100.095	00:03:52	2.102
222	0	0	00:03:53	0
223	0.021	100.095	00:03:54	2.102
224	0.021	100.095	00:03:55	2.102
225	0.062	33.903	00:03:56	2.102
226	0.021	100.095	00:03:57	2.102
227	0.022	95.545006	00:03:58	2.102
228	0.011	191.091	00:03:59	2.102
229	0.021	100.095	00:04:00	2.102
230	0.021	100.095	00:04:01	2.102
231	0.024	87.58301	00:04:02	2.102
232	0.023	91.39101	00:04:03	2.102
233	0.024	87.58301	00:04:04	2.102
234	0.023	91.39101	00:04:05	2.102
235	0.022	95.545006	00:04:06	2.102
236	0.011	191.091	00:04:07	2.102
237	0.021	100.095	00:04:08	2.102
238	0.021	100.095	00:04:09	2.102
239	0.021	100.095	00:04:10	2.102
240	0.021	100.095	00:04:11	2.102
241	0.061	34.459003	00:04:12	2.102
242	0.021	100.095	00:04:13	2.102
243	0.021	100.095	00:04:14	2.102
244	0.011	191.091	00:04:15	2.102
245	0.021	100.095	00:04:16	2.102
246	0.02	105.100006	00:04:17	2.102
247	0.021	100.095	00:04:18	2.102
248	0.021	100.095	00:04:19	2.102
249	0.021	100.095	00:04:20	2.102
250	0.021	100.095	00:04:21	2.102
251	0.021	100.095	00:04:22	2.102
251	0	0	00:04:23	0
252	0.011	191.091	00:04:24	2.102
253	0.021	100.095	00:04:25	2.102
254	0.021	100.095	00:04:26	2.102
255	0.021	100.095	00:04:27	2.102
256	0.061	34.459003	00:04:28	2.102
257	0.021	100.095	00:04:29	2.102
258	0.022	95.545006	00:04:30	2.102
259	0.021	100.095	00:04:31	2.102
260	0.022	95.545006	00:04:32	2.102
261	0.021	100.095	00:04:33	2.102
262	0.012	175.167	00:04:34	2.102
263	0.021	100.095	00:04:35	2.102
264	0.013	161.692	00:04:36	2.102
265	0.021	100.095	00:04:37	2.102
266	0.023	91.39101	00:04:38	2.102
267	0.021	100.095	00:04:39	2.102
268	0.021	100.095	00:04:40	2.102
269	0.021	100.095	00:04:41	2.102
270	0.021	100.095	00:04:42	2.102
271	0.037	56.811	00:04:43	2.102
272	0.046	45.696003	00:04:44	2.102
273	0.022	95.545006	00:04:45	2.102
274	0.021	100.095	00:04:46	2.102
275	0.022	95.545006	00:04:47	2.102
276	0.021	100.095	00:04:48	2.102
277	0.022	95.545006	00:04:49	2.102
278	0.021	100.095	00:04:50	2.102
279	0.022	95.545006	00:04:51	2.102
279	0	0	00:04:52	0
280	0.021	100.095	00:04:53	2.102
281	0.028	75.07101	00:04:54	2.102
282	0.021	100.095	00:04:55	2.102
283	0.022	95.545006	00:04:56	2.102
284	0.021	100.095	00:04:57	2.102
285	0.022	95.545006	00:04:58	2.102
286	0.061	34.459003	00:04:59	2.102
287	0.021	100.095	00:05:00	2.102
288	0.021	100.095	00:05:01	2.102
289	0.021	100.095	00:05:02	2.102
290	0.021	100.095	00:05:03	2.102
291	0.021	100.095	00:05:04	2.102
292	0.011	191.091	00:05:05	2.102
293	0.021	100.095	00:05:06	2.102
294	0.022	95.545006	00:05:07	2.102
295	0.021	100.095	00:05:08	2.102
296	0.021	100.095	00:05:09	2.102
297	0.021	100.095	00:05:10	2.102
298	0.022	95.545006	00:05:11	2.102
299	0.021	100.095	00:05:12	2.102
300	0.011	191.091	00:05:13	2.102
301	0.021	100.095	00:05:14	2.102
302	0.021	100.095	00:05:15	2.102
303	0.036	58.389004	00:05:16	2.102
304	0.031	67.806	00:05:17	2.102
305	0.022	95.545006	00:05:18	2.102
306	0.022	95.545006	00:05:19	2.102
307	0.021	100.095	00:05:20	2.102
308	0.012	175.167	00:05:21	2.102
308	0	0	00:05:22	0
309	0.021	100.095	00:05:23	2.102
310	0.021	100.095	00:05:24	2.102
311	0.022	95.545006	00:05:25	2.102
312	0.021	100.095	00:05:26	2.102
313	0.021	100.095	00:05:27	2.102
314	0.022	95.545006	00:05:28	2.102
315	0.021	100.095	00:05:29	2.102
316	0.012	175.167	00:05:30	2.102
317	0.021	100.095	00:05:31	2.102
318	0.021	100.095	00:05:32	2.102
319	0.061	34.459003	00:05:33	2.102
320	0.016	131.375	00:05:34	2.102
321	0.021	100.095	00:05:35	2.102
322	0.021	100.095	00:05:36	2.102
323	0.021	100.095	00:05:37	2.102
324	0.011	191.091	00:05:38	2.102
325	0.021	100.095	00:05:39	2.102
326	0.021	100.095	00:05:40	2.102
327	0.021	100.095	00:05:41	2.102
328	0.021	100.095	00:05:42	2.102
329	0.021	100.095	00:05:43	2.102
330	0.021	100.095	00:05:44	2.102
331	0.021	100.095	00:05:45	2.102
332	0.021	100.095	00:05:46	2.102
333	0.021	100.095	00:05:47	2.102
334	0.061	34.459003	00:05:48	2.102
335	0.022	95.545006	00:05:49	2.102
336	0.021	100.095	00:05:50	2.102
336	0	0	00:05:51	0
337	0.021	100.095	00:05:52	2.102
338	0.021	100.095	00:05:53	2.102
339	0.022	95.545006	00:05:54	2.102
340	0.011	191.091	00:05:55	2.102
341	0.022	95.545006	00:05:56	2.102
342	0.02	105.100006	00:05:57	2.102
343	0.021	100.095	00:05:58	2.102
344	0.021	100.095	00:05:59	2.102
345	0.021	100.095	00:06:00	2.102
346	0.021	100.095	00:06:01	2.102
347	0.021	100.095	00:06:02	2.102
348	0.022	95.545006	00:06:03	2.102
349	0.021	100.095	00:06:04	2.102
350	0.022	95.545006	00:06:05	2.102
351	0.02	105.100006	00:06:06	2.102
352	0.022	95.545006	00:06:07	2.102
353	0.021	100.095	00:06:08	2.102
354	0.022	95.545006	00:06:09	2.102
355	0.021	100.095	00:06:10	2.102
356	0.021	100.095	00:06:11	2.102
357	0.021	100.095	00:06:12	2.102
358	0.021	100.095	00:06:13	2.102
359	0.021	100.095	00:06:14	2.102
359	0	0	00:06:15	0
360	0.037	56.811	00:06:16	2.102
361	0.041	51.268	00:06:17	2.102
362	0.021	100.095	00:06:18	2.102
363	0.021	100.095	00:06:19	2.102
364	0.011	191.091	00:06:20	2.102
365	0.021	100.095	00:06:21	2.102
366	0.021	100.095	00:06:22	2.102
367	0.02	105.100006	00:06:23	2.102
368	0.021	100.095	00:06:24	2.102
369	0.021	100.095	00:06:25	2.102
370	0.021	100.095	00:06:26	2.102
371	0.021	100.095	00:06:27	2.102
372	0.011	191.091	00:06:28	2.102
373	0.022	95.545006	00:06:29	2.102
374	0.021	100.095	00:06:30	2.102
375	0.021	100.095	00:06:31	2.102
376	0.07600001	27.658	00:06:32	2.102
377	0.022	95.545006	00:06:33	2.102
378	0.021	100.095	00:06:34	2.102
379	0.022	95.545006	00:06:35	2.102
380	0.012	175.167	00:06:36	2.102
381	0.022	95.545006	00:06:37	2.102
382	0.021	100.095	00:06:38	2.102
383	0.021	100.095	00:06:39	2.102
384	0.021	100.095	00:06:40	2.102
385	0.021	100.095	00:06:41	2.102
386	0.021	100.095	00:06:42	2.102
387	0.021	100.095	00:06:43	2.102
387	0	0	00:06:44	0
388	0.021	100.095	00:06:45	2.102
389	0.021	100.095	00:06:46	2.102
390	0.037	56.811	00:06:47	2.102
391	0.031	67.806	00:06:48	2.102
392	0.021	100.095	00:06:49	2.102
393	0.021	100.095	00:06:50	2.102
394	0.021	100.095	00:06:51	2.102
395	0.021	100.095	00:06:52	2.102
396	0.011	191.091	00:06:53	2.102
397	0.021	100.095	00:06:54	2.102
398	0.022	95.545006	00:06:55	2.102
399	0.021	100.095	00:06:56	2.102
400	0.022	95.545006	00:06:57	2.102
401	0.021	100.095	00:06:58	2.102
402	0.022	95.545006	00:06:59	2.102
403	0.022	95.545006	00:07:00	2.102
404	0.012	175.167	00:07:01	2.102
405	0.053	39.660004	00:07:02	2.102
406	0.021	100.095	00:07:03	2.102
407	0.021	100.095	00:07:04	2.102
408	0.021	100.095	00:07:05	2.102
409	0.022	95.545006	00:07:06	2.102
410	0.021	100.095	00:07:07	2.102
411	0.021	100.095	00:07:08	2.102
412	0.021	100.095	00:07:09	2.102
413	0.021	100.095	00:07:10	2.102
413	0	0	00:07:11	0
414	0.021	100.095	00:07:12	2.102
415	0.021	100.095	00:07:13	2.102
416	0.021	100.095	00:07:14	2.102
417	0.04	52.550003	00:07:15	2.102
418	0.036	58.389004	00:07:16	2.102
419	0.041	51.268	00:07:17	2.102
420	0.011	191.091	00:07:18	2.102
421	0.022	95.545006	00:07:19	2.102
422	0.021	100.095	00:07:20	2.102
423	0.021	100.095	00:07:21	2.102

424	0.021	100.095	00:07:22	2.102	478	0.015	140.13301	00:08:18	2.102
425	0.021	100.095	00:07:23	2.102	479	0.021	100.095	00:08:19	2.102
426	0.021	100.095	00:07:24	2.102	480	0.022	95.545006	00:08:20	2.102
427	0.022	95.545006	00:07:25	2.102	481	0.021	100.095	00:08:21	2.102
428	0.02	105.100006	00:07:26	2.102	482	0.022	95.545006	00:08:22	2.102
429	0.021	100.095	00:07:27	2.102	483	0.011	191.091	00:08:23	2.102
430	0.021	100.095	00:07:28	2.102	484	0.022	95.545006	00:08:24	2.102
431	0.021	100.095	00:07:29	2.102	485	0.021	100.095	00:08:25	2.102
432	0.021	100.095	00:07:30	2.102	486	0.022	95.545006	00:08:26	2.102
433	0.037	56.811	00:07:31	2.102	487	0.021	100.095	00:08:27	2.102
434	0.031	67.806	00:07:32	2.102	488	0.022	95.545006	00:08:28	2.102
435	0.011	191.091	00:07:33	2.102	489	0.011	191.091	00:08:29	2.102
436	0.027	77.852005	00:07:34	2.102	490	0.022	95.545006	00:08:30	2.102
437	0.021	100.095	00:07:35	2.102	491	0.021	100.095	00:08:31	2.102
438	0.021	100.095	00:07:36	2.102	492	0.022	95.545006	00:08:32	2.102
439	0.021	100.095	00:07:37	2.102	493	0.071	29.606	00:08:33	2.102
440	0.022	95.545006	00:07:38	2.102	494	0.021	100.095	00:08:34	2.102
441	0.011	191.091	00:07:39	2.102	494	0	0	00:08:35	0
441	0	0	00:07:40	0	495	0.021	100.095	00:08:36	2.102
442	0.022	95.545006	00:07:41	2.102	496	0.025	84.08	00:08:37	2.102
443	0.021	100.095	00:07:42	2.102	497	0.021	100.095	00:08:38	2.102
444	0.022	95.545006	00:07:43	2.102	498	0.021	100.095	00:08:39	2.102
445	0.021	100.095	00:07:44	2.102	499	0.022	95.545006	00:08:40	2.102
446	0.022	95.545006	00:07:45	2.102	500	0.021	100.095	00:08:41	2.102
447	0.021	100.095	00:07:46	2.102	501	0.042	50.048004	00:08:42	2.102
448	0.037	56.811	00:07:47	2.102	502	0.021	100.095	00:08:43	2.102
449	0.041	51.268	00:07:48	2.102	503	0.021	100.095	00:08:44	2.102
450	0.021	100.095	00:07:49	2.102	504	0.021	100.095	00:08:45	2.102
451	0.011	191.091	00:07:50	2.102	505	0.022	95.545006	00:08:46	2.102
452	0.021	100.095	00:07:51	2.102	506	0.021	100.095	00:08:47	2.102
453	0.021	100.095	00:07:52	2.102	507	0.062	33.903	00:08:48	2.102
454	0.021	100.095	00:07:53	2.102	508	0.021	100.095	00:08:49	2.102
455	0.021	100.095	00:07:54	2.102	509	0.021	100.095	00:08:50	2.102
456	0.021	100.095	00:07:55	2.102	510	0.021	100.095	00:08:51	2.102
457	0.011	191.091	00:07:56	2.102	511	0.022	95.545006	00:08:52	2.102
458	0.021	100.095	00:07:57	2.102	512	0.021	100.095	00:08:53	2.102
459	0.021	100.095	00:07:58	2.102	513	0.022	95.545006	00:08:54	2.102
460	0.022	95.545006	00:07:59	2.102	514	0.021	100.095	00:08:55	2.102
461	0.02	105.100006	00:08:00	2.102	515	0.021	100.095	00:08:56	2.102
462	0.021	100.095	00:08:01	2.102	516	0.011	191.091	00:08:57	2.102
463	0.046	45.696003	00:08:02	2.102	517	0.021	100.095	00:08:58	2.102
464	0.023	91.39101	00:08:03	2.102	518	0.021	100.095	00:08:59	2.102
465	0.014	150.143	00:08:04	2.102	519	0.021	100.095	00:09:00	2.102
465	0	0	00:08:05	0	520	0.021	100.095	00:09:01	2.102
466	0.021	100.095	00:08:06	2.102	520	0	0	00:09:02	0
467	0.012	175.167	00:08:07	2.102	521	0.059	35.627003	00:09:03	2.102
468	0.021	100.095	00:08:08	2.102	522	0.021	100.095	00:09:04	2.102
469	0.021	100.095	00:08:09	2.102	523	0.021	100.095	00:09:05	2.102
470	0.021	100.095	00:08:10	2.102	524	0.022	95.545006	00:09:06	2.102
471	0.021	100.095	00:08:11	2.102	525	0.021	100.095	00:09:07	2.102
472	0.021	100.095	00:08:12	2.102	526	0.02	105.100006	00:09:08	2.102
473	0.011	191.091	00:08:13	2.102	527	0.021	100.095	00:09:09	2.102
474	0.021	100.095	00:08:14	2.102	528	0.012	175.167	00:09:10	2.102
475	0.021	100.095	00:08:15	2.102	529	0.022	95.545006	00:09:11	2.102
476	0.021	100.095	00:08:16	2.102	530	0.022	95.545006	00:09:12	2.102
477	0.037	56.811	00:08:17	2.102	531	0.021	100.095	00:09:13	2.102
532	0.022	95.545006	00:09:14	2.102					
533	0.021	100.095	00:09:15	2.102					
534	0.021	100.095	00:09:16	2.102					
535	0.061	34.459003	00:09:17	2.102					
536	0.021	100.095	00:09:18	2.102					
537	0.021	100.095	00:09:19	2.102					
538	0.021	100.095	00:09:20	2.102					
539	0.021	100.095	00:09:21	2.102					
540	0.021	100.095	00:09:22	2.102					
541	0.021	100.095	00:09:23	2.102					
542	0.011	191.091	00:09:24	2.102					
543	0.021	100.095	00:09:25	2.102					
544	0.021	100.095	00:09:26	2.102					
545	0.021	100.095	00:09:27	2.102					
546	0.021	100.095	00:09:28	2.102					
547	0.021	100.095	00:09:29	2.102					
547	0	0	00:09:30	0					
548	0.011	191.091	00:09:31	2.102					
549	0.021	100.095	00:09:32	2.102					
550	0.071	29.606	00:09:33	2.102					
551	0.022	95.545006	00:09:34	2.102					
552	0.02	105.100006	00:09:35	2.102					
553	0.022	95.545006	00:09:36	2.102					
554	0.025	84.08	00:09:37	2.102					
555	0.022	95.545006	00:09:38	2.102					
556	0.021	100.095	00:09:39	2.102					
557	0.021	100.095	00:09:40	2.102					
558	0.011	191.091	00:09:41	2.102					
559	0.021	100.095	00:09:42	2.102					
560	0.011	191.091	00:09:43	2.102					
561	0.021	100.095	00:09:44	2.102					
562	0.022	95.545006	00:09:45	2.102					
563	0.021	100.095	00:09:46	2.102					
564	0.011	191.091	00:09:47	2.102					
565	0.036	58.389004	00:09:48	2.102					
566	0.031	67.806	00:09:49	2.102					
567	0.021	100.095	00:09:50	2.102					
568	0.022	95.545006	00:09:51	2.102					
569	0.021	100.095	00:09:52	2.102					
570	0.021	100.095	00:09:53	2.102					
571	0.021	100.095	00:09:54	2.102					
572	0.021	100.095	00:09:55	2.102					
573	0.021	100.095	00:09:56	2.102					
574	0.012	175.167	00:09:57	2.102					
575	0.021	100.095	00:09:58	2.102					
576	0.012	175.167	00:09:59	2.102					
576	0	0	00:10:00	0					
577	0.021	100.095	00:10:01	2.102					
578	0.022	95.545006	00:10:02	2.102					
	13.191001	59036.5109							
Average	0.0228218	102.139292							

C.3 Needed Times to Learn the Techniques and Solve the Experimental Task

Metric 3d, time to become acquainted with the techniques in h		Metric 3e, time to solve the preparatory exercises in h	
CB-SPE	Palladio	CB-SPE	Palladio
3	2	3	3
2	1	3	3
3.5	3.5	5	5
0.25	0.17	7	5
2.5	0.8	6.5	3
1	1	4.5	2.5
5	1	4	3
3	0.5	6	2
2	1	5	2
4	1	11	5
2	2	4	4
1.5	0.3	16	3.5
2.5	1	6	4
0	0	10	5
2	3	10	10
1	1	2	2
1	1	4	2
3	1	4	3
4	0.75	8	3
4	2	2	2

Table 25: Needed Times to Learn the Techniques and Work on Preparatory Exercise (metrics 3d and 3e)

Time in minutes		
CB-SPE	Palladio	intuitive
155	180	90
150	210	40
120	135	40
140	120	30
120	135	
120	120	
120	90	
90	105	
78	120	
130	135	
	150	

Table 26: List of Needed Times for Experimental Exercise (Metric 3c)

Empirical Validation and Comparison of the Model-Driven Performance Prediction Techniques CB-SPE and Palladio

Declaration

I declare that this thesis is my own work and has not been submitted in any form to another university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

August 15th, 2005
Anne Martens