Karlsruhe Institute of Technology

Workload Classification and Forecasting

Diploma Thesis of

Nikolas Roman Herbst

At the Department of Informatics
Institute for Program Structures
and Data Organization (IPD)

Reviewer:	Prof. Dr. Ralf Reussner
Second reviewer:	Prof. Dr. Walter Tichy
Advisor:	Dipl. Inform. Nikolaus Huber
Second advisor:	Dist. Eng. Erich Amrehn (IBM)

Duration: March 1st, 2012 – August 31st, 2012

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, August 10th, 2012

.....

Nikolas Roman Herbst

Contents

Abstract - German

Virtualisierungstechnologien ermöglichen eine dynamische Zuordnung von Rechenressourcen an Ausführungsumgebungen zur Laufzeit. Um das mit diesem Freiheitsgrad verbundene Optimierungspotenial ausschöpfen zu können, sind zuverlässige Vorhersagen über die Intensität der einem System ankommenden Arbeitslast wertvolle Informationen, um kontinuierlich die Dienstgüte einer Anwendung sicherzustellen und zugleich die Effizienz der Ressourcennutzung zu steigern.

Die Zeitreihenanalyse bietet ein breites Spektrum an Methoden zur Berechnung von Vorhersagen, basierend auf periodisch beobachteten Erfahrungswerten an. Verwandte Arbeiten im Feld der proaktiven Ressourcenplanung konzentrieren sich meist auf einzelne Methoden der Zeitreihenanalyse und deren individuelles Optimierungspotential, so dass sich verwertbare Vorhersagen ausschließlich in bestimmten Situationen erzielen lassen.

In dieser Diplomarbeit werden die etablierten Methoden der Zeitreihenanalyse hinsichtlich ihrer Stärken und Schwächen analysiert und gruppiert. Ein Ansatz wird vorgestellt, der basierend auf einem Entscheidungsbaum und direktem Feedback über die Genauigkeit einer Vorhersage dynamisch die für die momentane Situation geeignete Methode auswählt. Ein Nutzer definiert lediglich seine generellen Zielsetzungen und Anforderungen an die Vorhersage.

Es wird eine Implementierung des vorgestellten theoretischen Ansatzes präsentiert, die Vorhersagen über die Intensität der ankommenden Arbeitslast kontinuierlich in konfigurierbaren Intervallen und unter kontrollierbarem Rechenaufwand zur Laufzeit liefert.

Basierend auf realen Arbeitslastverläufen werden eine Reihe an unterschiedlichen Experimenten und eine Fallstudien durchgeführt. Die Ergebnisse zeigen, dass es durch Einsatz der Implementierung im Vergleich zur statischen Anwendung einer etablierten Methode gelingt, den relativen Fehler der vorhergesagten Punkte in Bezug auf die eintretenden Beobachtungen im Mittel um 63% zu reduzieren. In einer Fallstudie werden durch proaktive Ressourcenplanung, basierend auf den Vorhersagen der Implementierung, zwischen 52% und 70% der normalerweise eintretenden Verletzungen einer fixen Leistungsvereinbarung verhindert.

Abstract - English

Virtualisation technologies enable dynamic allocation of computing resources to execution environments at run-time. To exploit optimisation potential that comes with these degrees of freedom, forecasts of the arriving work's intensity are valuable information, to continuously ensure a defined quality of service (QoS) definition and at the same time to improve the efficiency of the resource utilisation.

Time series analysis offers a broad spectrum of methods for calculation of forecasts based on periodically monitored values. Related work in the field of proactive resource provisioning mostly concentrate on single methods of the time series analysis and their individual optimisation potential. This way, usable forecast results are achieved only in certain situations.

In this thesis, established methods of the time series analysis are surveyed and grouped concerning their strengths and weaknesses. A dynamic approach is presented that selects based on a decision tree and direct feedback cycles, capturing the forecast accuracy, the suitable method for a given situation. The user needs to provide only his general forecast objectives.

An implementation of the introduced theoretical approach is presented that continuously provides forecasts of the arriving work's intensity in configurable intervals and with controllable computational overhead during run-time.

Based on real-world intensity traces, a number of different experiments and a case study is conducted. The results show, that by use of the implementation the relative error of the forecast points in relation to the arriving observations is reduced by 63% in average compared to the results of a statically selected, sophisticated method. In a case study, between 52% and 70% of the violations of a given service level agreement are prevented by applying proactive resource provisioning based on the forecast results of the introduced implementation.

1. Introduction

The use of virtualisation technologies introduces an abstraction layer between resources that are physically available and resources that are logically visible to an execution environment hosted on top of the virtualization layer. Thereby, virtualization allows to dynamically assign and withdraw resources to and from hosted execution environments at run-time. The amount of resource consumed by executed software services are mainly induced by the arriving work's intensity and therefore are likely to change over the course of time. The flexibility in resource provisioning that comes with virtualisation and the variation over time in resource demands raise a dynamic optimisation problem. Mechanisms that continuously provide appropriate solutions to this optimisation problem could exploit the potential to use physical resources more efficiently, resulting in cost and energy savings. The challenges and potentials to reduce costs and the energy consumption by intelligent resource management using virtualization are discussed in many research papers [OAL11, AMM+09, TBL09, OLG10, LO10].

In general, mechanisms that try to continuously match the amounts of demanded to provisioned resources are reactive using threshold based rules to detect and react on resource shortages. However, such reactive mechanisms can be combined with a more intelligent one that proactively anticipates changes in resource demands and prevent shortages. To achieve this, forecasts of the arriving work's intensity are a crucial building block.

Related research work in the field of proactive resource provisioning as it can be found in [GRCK07, HMBN10, BKB07, MIK+10, KKK10, CDM11] mostly concentrates on single forecast methods of the time series analysis and their individual optimisation potential. This way, usable forecast results are achieved only in certain situations. In addition, the forecasts are applied in the majority of cases on monitored performance metrics that are depending on the recent combination of consumed and provisioned resources and therefore cannot directly quantify the amount of arriving work.

In this thesis, the broad spectrum of methods offered by the time series analysis to compute forecasts based on periodically monitored values is surveyed and the individual methods are grouped concerning their strengths, weaknesses and requirements. A dynamic approach is presented that selects at run-time the most suitable method for a given situation. This selection is based on a decision tree that captures forecast objectives, individual requirements and integrates heuristics, and direct feedback cycles on forecast accuracy. Users of the approach need to provide only their general forecast objectives and are not responsible to dynamically select appropriate forecast methods. An implementation of

the introduced theoretical approach is presented that continuously provides forecasts of the arriving work's intensity in configurable intervals and with controllable computational overhead during run-time.

The approach and the implementation are evaluated in multiple different experiments and a case study, all based on real-world arriving workload intensity traces. The results show that the intelligent and dynamic selection of a suitable forecast strategy reduces the relative error of the forecast points in relation to the arriving observations by 63% in average compared to the results of a statically selected, sophisticated method. Even when limiting the space of selectable forecast methods to a certain group of similar ones, the implementation achieves a higher forecast accuracy with less outliers than individual methods of the group in focus. In the conducted case study, between 52% and 70% of the violations of a given service level agreement are prevented by applying proactive resource provisioning based on the forecast results of the introduced implementation.

1.1 Goals of the Thesis

The contributions of this diploma thesis are formulated in the following goals that are tackled stepwise in the individual chapters.

Identify the characteristics and components of a workload intensity behavior and metrics to quantify them for automatic classification and selection of a forecast strategy.

Build a survey on state-of-the-art forecasting approaches (including interpolation, decomposition and pattern recognition techniques) of the time series analysis focusing on their strengths, weaknesses and requirements and analyse the computational overhead of the individual forecasting strategies to identify suitable configurations for their online application. Identify metrics that capture or estimate the accuracy of a forecast and enable direct comparison of different strategies.

Design an approach to automatically correlate workload timing behavior classes and time series based forecasting approaches using a decision tree that considers forecast objectives, incorporates feedback on the forecast accuracy and offers a configurable space for further heuristic optimisation.

Build a `WorkloadClassificationAndForecasting` system for online application that provides continuous forecast results of a variable number of different workload intensity behaviors. This goal includes known software engineering tasks like designing the system's architecture, specifying its interfaces, selecting appropriate technologies and implementation as well as testing and documentation.

Validate the introduced approach by conducting experiments at the `WorkloadClassificationAndForecasting` system and illustrate the value of its benefits in a case study.

Achieving these goals in combination can be seen as a step towards the Descartes Research Group's vision of a *Self-aware System* as outlined in [KBHR10], since an application of the resulting approach enables a system to analyse its past usage to intelligently anticipate recurring patterns or trends.

1.2 Thesis Structure

According to the stated main goals, the remainder of this thesis is structured as outlined in the following:

The foundations in Chapter 2 start with a section on definitions, terms and differentiations, before Section 2.2 describes characteristics of workload intensity behaviors that are captured in time series of request arrival rates. It further refines these characteristics in dimensions and corresponding metrics for classification. Section 2.3 gives short summary on pattern identification methods that are applicable to time series data and an survey on existing forecast approaches that use time series analysis This sections ends with a survey on forecast accuracy metrics. Section 2.4 outlines the application context in which forecast results can be used.

Chapter 3 presents the workload classification and forecasting approach and starts with a section on assumptions and limitations. Section 3.2 explains the individual concepts before they are combined in a decision tree with feedback cycles as outlined and illustrated in 3.3

Section 4.1 outlines the architecture of the `WorkloadClassificationAndForecasting` system that realises the presented approach. The following Section 4.2 captures implementation details on the individual system components. Section 4.3 discusses design decisions and variation or extension points, before the last Section 4.4 of this chapter gives technical details on the development and run-time environments.

The evaluation of the approach and its implementation can be found in Chapter 5 that starts with a section containing details on the used real-world workload traces. In Section 5.2, four experiments with different focus are presented and their results illustrated and explained, before the value of benefit that is connected to an application of the presented implementation is illustrated in a case study. In the last section of Chapter 5, the experiments' results are interpreted and discussed.

A discussion of related research work can be found at the end of this thesis in Chapter 6. before the concluding remarks and future prospects in Chapter 7.

2. Foundations

This chapter starts with a section on definitions of crucial terms from the fields of virtualisation, software services and workload characterisation as well as from the field of the performance analysis. Section 2.2 discusses the characteristics of workload intensity behaviors and outlines ways to quantify them. The following section starts by giving a brief overview on pattern identification methods offered by the time series analysis that are already build in or can be combined with forecast methods. Subsection 2.3.2 gives a survey on state-of-the-art forecast methods and analyses them concerning their strengths, weaknesses and requirements. This section ends with a summary of metrics that are able to quantify the forecast accuracy and enable direct comparisons of different forecast methods. The last section of this chapter outlines the application context for which the approach is designed as presented in Chapter 3.

2.1 Definitions, Terms and Differentiations

The aim of this section is to enable the reader to build up a precise understanding of the presented concepts and ideas as well as of the thesis' context. Crucial terms concerning virtualisation, software services, workload characterisation and performance analysis are lined up and defined for the context of this thesis. The definitions can also be found in the glossary at the end of this thesis.

The following four items define terms in the context of virtualisation, as this technology offers functionality that enables dynamic resource provisioning.

An **execution platform** consists of hardware, virtualization and operating system layer plus optional middleware like an application server.

Scalability: A computing system consisting of an execution platform and applications is scalable if both properties are fulfilled:

- **Platform scalability** is the ability of the execution platform to provide and make use of as many (additional) resources as needed (or explicitly requested) by an application.
- **Application scalability** is a property of the software/application layers. The application is able to maintain its performance goals in terms of response time or throughput as defined in Service Level Agreements (SLA) even when its workload intensity increases. Platform scalability is a necessary property to make application scalability possible.

These properties imply the absence of bottlenecks in both active and passive resources. Scalability of a computing system can be limited by an upper bound of workload intensity or resource pool size. [KHvKR11]

Elasticity of a computing system is characterized by the temporal and quantitative properties of automated scaling, which is run-time resource provisioning and unprovisioning performed by the execution platform. A manually scaled system cannot be called elastic. Execution platform elasticity depends on the state of the platform and on the state of the platform-hosted applications. Elasticity implies Scalability beneath a given upper bound. [KHvKR11].

A **virtualized elastic system** is an execution platform that is elastic in terms of the given definition and makes use of virtualization technologies. By using virtualization technology the underlying physical resources can be mapped transparently and dynamically to virtual resources.

In the following, nine terms are defined concerning software services and the work that is processed by these software services:

Software services (SaaS) are offered by a computing system to the users of the system, which can be human persons via an interface or computing machines. In our context, a software service can be seen as a deployed software component.

Requests are submitted to a software service by a user and can be seen as an encapsulation of a single usage of a service.

A **request class** is a category of requests that is characterized by statistically indistinguishable resource demands.

A **resource demand** in units of time or capacity is the consumption of physical or virtual resources induced by processing a single request.

A **workload** is the physical (not modeled) usage of a system over time containing requests of one or more request classes. A workload can contain usage patterns that enable load forecasting, provisioning, planning or anomaly analysis. This definition deviates from the definition in [TL10] on page 2, where a workload is a more general term capturing applications and their service level agreements additionally.

Workload models are representations that capture the main aspects of a corresponding real workload that has effect on the performance measures of interest. [Kou07]

A **usage scenario** is an instance of a workload model and defines the rates and order of service requests and is used for early performance predictions in the context of the Palladio Component Model (PCM) as described in [RBH+07].

A **workload category** is a coarse-grained classification of a workload and divided into four basic application and technology domains broadly used for market segmentation and analysis. As in [TL10] on page 13 the four categories are:

- Database and Transaction Processing
- Business Process Applications
- Analytics and High Performance Computing
- Web Collaboration and Infrastructure

The last five items of this section focus on terms that are needed to capture characteristics of variable arriving work to a software service as well as on three basic performance metrics throughput, resource utilization and response time:

A **time series** X is a discrete function that represents real-valued measurements $x_i \in R$ for every time point t_i in a set of n equidistant time points $t = t_1, t_2, ..., t_n$: $X = x_1, x_2, ..., x_n$ as described in [Mit09]. A time series may have a finite capacity. The time between to time series points is defined by a value and a time unit. The number of time series points, that add up to a higher time unit or another obvious period is the frequency of a time series. The frequency attribute of a time series is an important start value for the search of seasonal patterns.

A **time series of request arrival rates** is a time series that contains $n_i \in N$ sums of unique request arrivals during the corresponding time intervals $[t_i, t_{i+1})$.

Workload intensity behavior is a description of a workload's characteristic changes in intensity over time like seasonal patterns, trends, noise, burstiness, level and more like positivity or burstiness. The workload intensity behavior can be extracted from a corresponding time series of request arrival rates.

Throughput is the rate at which requests are completed by a computer system (measured in operations per unit of time). [Kou05]

Resource utilization is the fraction of time that the resource is busy. [Kou05]

Response time is the time it takes a system to react to a request and is composed of congestion time(s) and service time(s). [Kou05]

2.2 Workload Intensity Behavior Characteristics

Already in 1985, Calzarossa et al. published a research paper [CS85] on characterization of the variation in time of workload arrival patterns, where they use polynomial regression to identify and describe daily workload patterns. In 1996, Arlitt and Carey searched for invariants in web-server workloads as they describe in their paper [AW96] focusing on characteristics of request classes and their typical resource demands. In the paper [SWHB06], the authors discuss and evaluate the two basic ways to model different workload types either in an open or a closed workload model. The generation of intensity varying workloads is captured in the publication [vHRH08], of Hoorn, Rohr and Hasselbring, where the influences on the intensity variation of a workload are modelled in detail to achieve the generation of realistic behaviors that can be used e.g. for benchmarking purposes.

The above listed research work on workload classification identifies central characteristics of workload intensity behaviors (WIB), but it is not in the focus to establish any connection between a certain characteristic and a suitable forecast strategy. To tackle the goal of building a classification scheme for a forecast strategy selection, it helps to decompose a workload intensity behavior in a first step. As a WIB is captured in a time series of request arrival rates, the components of the time series itself and its properties are central to the analysis.

2.2.1 Time Series Component Identification

According the theory of the time series analysis as presented in standard works of this field [BJR08, Hyn08, Shu11], a time series can be decomposed into the following three components. The relative weights and the shapes of these components characterise a corresponding workload intensity behavior.

The **trend** component can be described by a monotonic increasing or decreasing function (in most cases a linear function) that can be approximated using common regression techniques. A break within the trend component is caused by system extrinsic events and therefore cannot be forecast by analysis of historic observations but detected in

retrospect. It is possible to estimate the likelihood of a change within the trend component by analysis of the durations of historic trends.

The **season** component captures recurring patterns that are composed of at least one or more frequencies, for example daily, weekly or monthly patterns (but they do not need to be integer valued). These frequencies can be identified by using a Fast Fourier Transformation (FFT) or by auto-correlation techniques.

The **noise** component is an unpredictable overlay of various frequencies with different amplitudes changing quickly due to random influences on the time series. Noise can be reduced by applying smoothing techniques like weighted moving averages (WMA), by using lower sampling frequency or by a filter that eliminates high frequencies. A suitable tradeoff between reduction of noise and loss of information can enhance forecast accuracy.

A time series decomposition into the above mentioned components is illustrated in Figure 2.1. This decomposition of a time series has been presented in [VHZC10]. The authors offer an implementation of their approach for time series decomposition and detection of breaks in trends or seasonal components (BFAST)[1]. In the first row, the time series input data is plotted. The second row contains the detected (yearly) seasonal patterns, whereas the third row shows the estimated trends and several breaks within these trends. The remainder in the undermost row is difference between time series data and the sum of the trend and the seasonal component and can be seen as the non-deterministic noise component.

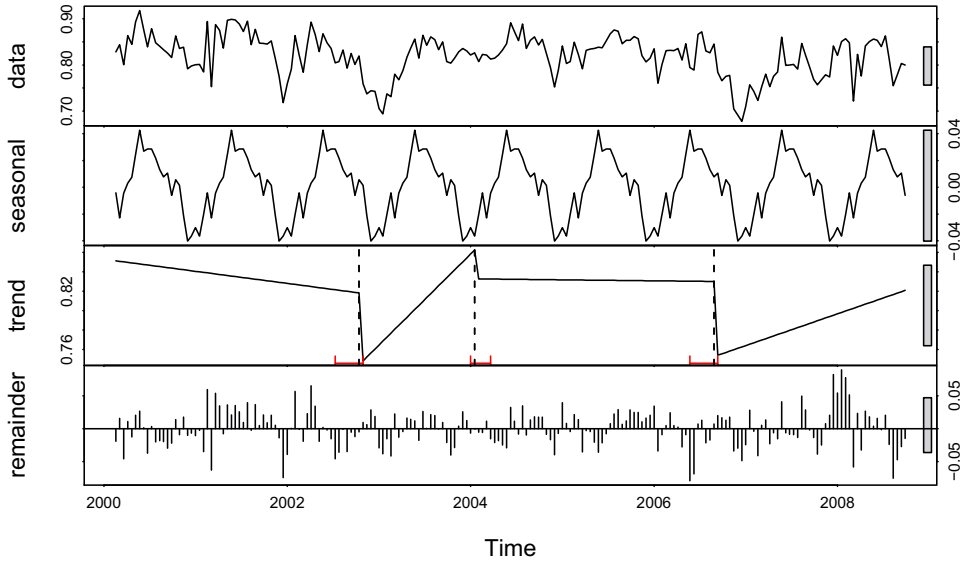


Figure 2.1: Illustration of a Time Series Decomposition into Season, Trend and Noise Components as presented in [VHZC10]

The theory of time series analysis differentiates between static and dynamic stochastic processes. In static process models it is assumed that the trend and season component stay constant. Having a dynamic process model, these components change or develop over time and therefore have to be approximated periodically to achieve good quality forecasts. Nevertheless, the trend and season components are considered as deterministic and the quality of their approximation is important for the quality of forecasts. Depending on the

[1] http://bfast.r-forge.r-project.org/

applied stochastic model, the season, trend and noise components can be either concerned as multiplicative or additive (or neglected) to build the original time series data.

2.2.2 Means for Automated Classification

As the decomposition of a time series using the BFAST approach induces a high computational burden, more easily computable characteristics of a time series are important for an online classification and selection of a suitable forecasting strategy. These metrics are presented in Table 2.1 in addition to the enumeration with short explanations below:

The *burstiness* index is a measure for the spontaneity of developments within the time series and calculated by the ratio of the maximum observed value to the median within a sliding window.

The *length* of the time series data mainly influences the accuracy of approximations for the above mentioned components.

The number of *consecutive monotonic values* either upwards or downwards within an sliding window characterises a time series as this value describes indirectly the influence of the noise and seasonal components on the time series. Observing a small value can be seen as a hint to apply a time series smoothing technique.

The *maximum*, the *median* and the *quartiles* are important to describe the distribution of time series values and can be unified in the *quartile dispersion coefficient* (QDC) as defined in common statistics as the distance of the quartiles divided by the median value.

The standard deviation and the mean value are combined in the *variance coefficient* that expresses unit-free the distribution of the time series values as the QDC.

Absolute *positivity* of a time series is an important characteristic, because intervals containing negative or zero values can influence the forecast quality and even the applicability of forecast strategies, as they can loose their numeric stability through the possibility of a division by zero. As arrival rates cannot be negative by nature but only zero, a not absolutely positive time series can be treated with a simple filter or a specialized forecasting strategy. Therefore, it is useful to compute the *rate of zero values* within a sliding window.

The application of *Weighted Moving Averages* is a simple approach to reduce noise effects in a time series without changing the level of data aggregation.

The *relative gradient* is defined in this context as the absolute gradient of the latest quarter of a time series period in relation the median of this quarter. It captures the steepness of the latest quarter period without a unit, as it is related to the median. A positive relative gradient shows that the last quarter period changed less than the median value, a negative value indicates a steep section within the time series (e.g. the limb of a seasonal component).

The *period* and *frequency* are properties of a time series and describe the number of time series values (frequency) that form a period of interest (in most cases simply the next bigger time-unit) and an important input as they are taken as starting values for the search of seasonal patterns.

These metrics introduce only low complexity in their computation, are sufficient to capture important characteristics of a workload intensity behavior and therefore are suitable to be integrated into an online classification process.

Table 2.1: Means for Characterisation and Classification of Workload Intensity Behaviors

Name	Calculation	Expressiveness
BurstinessIndex	The *BurstinessIndex* is defined as the maximum values divided by the median. A burst is a positive peak.	The *BurstinessIndex* characterizes the level of bursts (not the frequency of bursts) whereas an index value of close to zero stands for high level of bursts – no bursts if the index value is 1 or close.
Length	The recent *Length* of a time series is a property that quantifies the historic knowledge.	If the *Length* of the time series is smaller than certain thresholds, this factor limits the choices of forecast strategies, as their requirements differ in the minimum amount of time series data.
LongestMonotonic Section	By iteration through the time series values the number of consecutive monotonic values (up or downwards) is counted. The *LongestMonotonicSection* is the maximum of these numbers and can be related to the *Length* of the analyzed time series span.	The *LongestMonotonicSection* in relation the length is a rate that characterized the trend behavior of the time series. A rate close to zero indicates a high level of noise or high frequency seasonal components. A rate close to 1 indicates the absence of noise or seasonal components – in this case even naïve forecast strategies can have results of good quality.
Maximum	Search for the *Maximum* value in the set of values of the analyzed time series span.	The *Maximum* corresponds to level of the maximal burst in the analyzed time span.
Median	Sort the set of values of the analyzed time series span. The *Median* is the value in the middle (or the arithmetic mean of the two middle values).	The *Median* is an outlier tolerant measure that quantifies the base level of a time series over the analyzed time span.
Period and Frequency	*Period* and *Frequency* values are properties of the time series: The *Frequency* value defines the amount of time series points that cover a period. A period is often chosen as the next bigger time unit or corresponds to the duration of a known seasonal pattern.	If the time series *Length* is below one *Period* (and below 5), only naïve forecast strategies should be applied. Complex forecast strategies that cover seasonal components, need at least a time series *Length* of 3 *Periods*.
Quartile	*Quartiles* are the *Medians* of the lower or the upper half of the sorted set of values.	The tuple of the lower and upper *Quartiles* characterizes outlier tolerant the span / interval that covers 50 % of the values.
QuartileDispersion Coefficient (QDC)	The *QuartileDispersionCoefficient* is defined as the distance of the *Quartiles* divided by the *Median* value.	The *QuartileDispersionCoefficient* is another relative measure of dispersion. Again, a coefficient close to zero indicates low variance, a coefficient close or bigger than 1 indicates high variance.
RateOfZeroValues	The *RateOfZeroValues* is the number of zeros in relation to the analyzed length of the time series.	Most forecast strategies need non-zero values as input and therefore the zero values are removed before. If the rate is higher than a threshold one could assume an intermittent demand and apply a special forecast strategy (CROSTON)
RelativeGradient	The *RelativeGradient* is the absolute gradient of the latest quarter of a *Period* in relation the *Median*.	The *RelativeGradient* captures the steepness of the latest quarter *Period* without a unit, as it is related to the *Median* – a positive *RelativeGradient* shows that the last quarter *Period* changed less than the *Median* value, a negative value indicates a steep section within the time series (e.g. the limb of a seasonal component). In this case the choice of simple forecast strategies should be limited.
StandardDeviation	The *StandardDeviation* is the square root of the sample variance. The sample variance is the sum of squared distances to the mean value in relation to the degrees of freedom. The analyzed set should contain at least 30 values.	The *StandardDeviation* quantifies the average distance to the mean value and is therefore a measure of dispersion with a unit.
Variance Coefficient	The *VarianceCoefficient* is defined as the *StandardDeviation* divided by the mean value.	The *VarianceCoefficient* is a relative measure of dispersion (without a unit to enable comparison). A coefficient close to zero indicates low variance, a coefficient close or bigger than 1 indicates high variance.
WeightedMoving Averages (WMA)	*WeightedMovingAverages* are applied in a sliding window of a given window size and a weight vector to calculate a weighted average for every time series point. (E.g. for careful noise reduction: window size = 3 and weight vector (0.25, 0.5, 0.25)).	*WeightedMovingAverages* are a simple technique to smooth out noise components or bursts and not losing too much information on deterministic components as trend and seasonal patterns.

2.2.3 Real-World Workload Intensity Behavior

As software services are normally used directly or indirectly by human users, real-world workload intensity behavior traces are likely to show strong seasonal patterns in daily periods that are possibly overlaid by far longer periods like weeks or months. The shape of a daily seasonal pattern characterises a workload intensity behavior as these patterns are directly influenced by common human habits like working hours, lunch time and of course common sleeping hours. The calender that defines working days, weekends and holidays may also have an strong impact on the workload intensity behavior. Even the weather can possibly influence human usage behavior.

A high weight of a trend component within a workload intensity behavior trace is rarely seen at the scale of hours and days, but may be found in aggregated data at the scale of weeks, month or years for long term forecasts. If monitoring values are available in a high resolution at the scale of minutes or seconds, stronger trends may also be visible if the noise level is not to strong.

2.3 Time Series Analysis

This section starts with a short summary of pattern identification methods and lists recent research publications in this field of the time series analysis, before a survey on forecast approaches is conducted in 2.3.2. At the end of this section, metrics are presented that capture forecast accuracy.

2.3.1 Pattern Identification in Time Series

It is important to understand the past when trying to anticipate the future developments. Therefore it is necessary to extract the deterministic patterns that occur in the time series history as precise as possible to apply them in a forecast approach or for time series decomposition. Common methods for pattern identification of the time series analysis are lined up in the following:

The **Discrete Fourier Transformation** (DFT) analyses the frequencies within a time series. Frequencies that are found more often than others hint towards seasonal patterns within the time series with a duration of the detected frequency. DFT is for example integrated in the tBATS innovation state space modelling framework [DLHS11] as stated in 2.3.2.8 or used in approaches of related work like in [GRCK07, HMBN10].

Auto-correlation techniques highlight similarities of a time series within itself for a given lag that can be provided by a DFT analysis. If this lag is equal to the duration of a seasonal pattern, the auto-correlation function identifies seasonal patterns and measures their similarity. For example, this technique is applied in combination with DFT in [BKB07] as presented in Related Work 6.

Polynomial Regression is a method that fits a polynomial of given degree through given discrete points minimizing squared distances. This technique has been used for the description of daily patterns (know frequency) in the early work in 1985 of Calzarossa and Serazzi [CS85]. For a small polynomial degree, this method does not induce a high processing complexity and does not need an extensive history, but this regression technique is sensitive to noise. Linear regression is a common approach to extrapolate the trend component of a time series.

Pattern identification approaches are already integrated into or can be combined with forecast methods that consider seasonal components and are therefore crucial in wider

context of time series based forecasting. Still, they do not support WIB classification for forecast strategy selection and are therefore not in the focus of the workload classification and forecast approach and its implementation that relies on accuracy feedback and simple heuristics to limit the computational overhead.

Further advanced pattern identification approaches are presented and discussed in a broad spectrum of research publications. In Olszewski's PhD thesis [Ols01] a generalized feature extraction approach is introduced for structural pattern recognition in time series data. Warren and Liao present a survey on clustering approaches for time series data in [WL05]. The authors of the publication [ZLDL11] present a novel clustering approach in time series data. In 1992, Raatikainen published a paper [Raa93] where he critically assesses the use of the k-means algorithm for workload classification. In [SGL+11] the authors propose an approach for pattern identification and classification in multivariate time series. Such an approach could be applied to analyse and detect correlations between workload intensity behaviors of different request classes. The authors of the publications [ALSS95] and [HB10] focus in both cases on the detection of fuzzy patterns under scaling or translations or in the presence of noise. As demonstrated in [BBR+07, Kle], neuronal nets in combination with machine learning techniques can be used to detect patterns in time series data.

2.3.2 Survey on Forecast Approaches

This section lines up forecast approaches of the time series analysis and highlights their requirements, strengths and weaknesses together with a short summary that is based on information from the sources in [BJR08, Hyn08, HK08, Shu11]. All of the following forecast methods have been implemented by Hyndman in the R forecast package [2] and documented in his publication [HK08]. The implemented forecast methods are based either on the state-space approach as discussed in [Hyn08] or the ARIMA (auto-regressive integrated moving averages) approach for stochastic process modelling that is extensively presented in the book of Box, Jenkins and Reinsel [BJR08]. These two general stochastic process modelling frameworks overlap, but are not identical as in both cases there exist model instances that have no counterpart in the other and both have different strength and weaknesses as stated in the comparison part in [HK08].

The below listed and shortly summarized forecast approaches are selected to cover the state-of-the-art spectrum of the time series analysis for different objectives, data characteristics and are ordered in their computational complexity. Capturing the complexity in common O-notations is hardly possible besides in the two first simple cases. The time series size is only one part of the problem description, as in addition, the shape of seasonal patterns contained in the data and optimisation thresholds during a model fitting procedure strongly influence the processing times. Therefore, the computational complexity of the individual methods has been evaluated in experiments that contain a representative amount of forecast method executions in an environment as described in Section 4.4.

The first two presented methods below are very basic ones, whereas the third to seventh method include capability to estimate trends within a time series development and can be seen as instances of more extensive stochastic proves modelling frameworks. The last three forecast methods are stochastic proves modelling frameworks and capable to additionally model seasonal patterns in different peculiarity. Table 2.2 summarises the presented forecast methods by shortly covering their operating mode, giving information on a suitable forecast horizon, requirements and experimentally evaluated computational overheads as well as by depicting strengths, weaknesses and an optimal application scenario. Some more forecast approaches, like Holt-Winters approach as described in [Goo10] from 1960 or the Theta method are covered in the presented list as a special case of exponential smoothing.

[2] http://robjhyndman.com/software/forecast/

Table 2.2: Table of Forecast Strategies and their Properties

Name	Operating Mode	Forecast Horizon	Requirements	Overhead	Strengths	Weaknesses	Optimal Scenario
Naïve Forecast	The naïve forecast considers only the value of the most recent observation assuming that this value has the highest probability for the next forecast point.	**very short** term forecast (1-2 points)	single observation	nearly none O(1)	no historic data required, reference method	naïve -> no value for proactive provisioning	constant arrival rates
Moving Average (MA)	Calculation of an arithmetic mean within a sliding window of the x most recent observations.	**very short** term forecast (1-2 points)	two observations	very low O(log(n))	simplicity	sensitive to trends, seasonal component	constant arrival rates with low white noise
Simple Exponential Smoothing (SES)	Generalization of MA by using weights according to the exponential function to give higher weight to more recent values. 1st step: estimation of parameters for weights/exp. function 2nd step: calculation of weighted averages as point forecasts	**short** term forecast (< 5 points)	small number of historic observations (> 5)	experiment: below **80ms** for less than **100 values**	more flexible reaction on trends or other developments than MA	no season component modeled, only damping and no interpolation	time series with some noise and changes within trend, but no seasonal behavior
Cubic Smoothing Splines (CS)	Cubic splines are fitted to the univariate time series data to obtain a trend estimate and linear forecast function. Prediction intervals are constructed by use of a likelihood approach for estimation of smoothing parameters. The cubic splines method can be mapped to an ARIMA 022 stochastic process model with a restricted parameter space.	**short** term forecast (< 5 points)	small number of historic observations (> 5)	experiment: below **100ms** for less than **30 values** (more values do not sig. improve accuracy)	extrapolation of the trend by a linear forecast function	sensitive seasonal patterns (step edges), negative forecast values possible	strong trends, but minor seasonal behavior, low noise level
ARIMA 101 (auto-regressive integrated moving averages)	ARIMA 101 is an ARIMA stochastic process model instance parameterized with p = 1 as order of AR(p) process, d = 0 as order of integration, q = 1 as order of MA(q) process. In this case a stationary stochastic process is assumed (no integration) and no seasonality considered.	**short** term forecast (< 5 points)	small number of historic observations (> 5)	experiment: below **70ms** for less than **100 values**	trend estimation, (more careful than CS), fast	no season component in modeled, only positive time series	time series with some noise and changes within trend, but no seasonal behavior
Croston's Method (intermittent demand forecasting)	Decomposition of the time series that contains zero values into two separate sequences: a non-zero valued time series and a second that contains the time intervals of zero values. Independent forecast using SES and combination of the two independent forecasts. No confidence intervals are computed due to no consistent underlying stochastic model.	**short** term forecast (< 5 points)	small number of historic observations, containing zero values (> 10)	experiment: below **avg. 100 ms** for less than **100 values**	decomposition of zero-valued time series	no season component, no confidence intervals due to no underlying stochastic model	zero valued periods, active periods with trends, no strong seasonal comp., low noise
Extended Exponential Smoothing (ETS) (Innovation state space stochastic model framework)	1st step: model estimation: noise, trend and season components are either additive (A), or multiplicative (M) or not modeled (N) 2nd step: estimation of parameters for an explicit noise, trend and seasonal components 3rd step: calculation of point forecasts for level, trend and season components independently using SES and combination of results	**medium to long** term forecast (> 30 points)	at least 3 periods in time series data with an adequate frequency of [10;65]	experiment: up to **15 seconds** for less than **200 values,** high variability in computation time	capturing explicit noise, trend and season component in a multiplicative or additive innovation state space model	only positive time series	times series with clear and simple trend and seasonal component, moderate noise level
tBATS (trigonometric seasonal model, Box-Cox transformation, ARMA errors, trend + seasonal components)	The tBATS stochastic process modeling framework of innovations state space approach focuses modeling of complex seasonal time series (multiple/high frequency/non-integer seasonality) and uses Box-Cox transformation, Fourier representations with time varying coefficients and ARMA error correction. Trigonometric formulation of complex seasonal time series patterns to enable their identification by FFT and time series decomposition. Improved computational overhead using a new method for maximum-likelihood estimations.	**medium to long** term forecast (> 5 points)	at least 3 periods in time series data with an adequate frequency of [10;65]	experiment: up to **18 seconds** less than **200 values,** high variability in computation time	modeling capability of complex, non-integer and overlaying seasonal patterns and trends	only positive time series, complex process for time series modeling and decomposition	times series with one or more clear but possibly complex trend and seasonal components, only moderate noise level
ARIMA (auto-regressive integrated moving averages stochastic process model framework with seasonality)	The automated ARIMA model selection process of the R forecasting package starts with a complex estimation of an appropriate ARIMA$(p, d, q)(P, D, Q)_m$ model by using unit-root tests and an information criterions (like the AIC) in combination with a step-wise procedure for traversing a relevant model space. The selected ARIMA model is then fitted to the data to provide point forecasts and confidence intervals.	**medium to long** term forecast (> 5 points)	at least 3 periods in time series data with an adequate frequency of [10;65]	experiment: up to **50 seconds** for less than **200 values,** high variability in computation time	capturing noise, trend and season component in (multiplicative or additive) model, Achieves close confidence intervals	only positive time series, complex model estimation	times series with clear seasonal component (constant frequency), moderate noise level

2.3.2.1 Naive Forecast Method

The *naive* method has no inherent complexity as it takes the assumption that the last observed value is the most likely one to be observed in the next step. This method is usually taken as the reference method to enable the comparison of two other forecast approaches. It can be combined with a random-walk factor or a drift. This method induces no computational overhead beside the calculation of a confidence interval and requires only a single time series point to be applicable. The *naive* method is identical to the *arithmetic mean* method with a sliding window size of one.

2.3.2.2 Moving Averages Method

The *moving average* (MA) of a sliding window can be computed and used as a point forecast for the next interval. Compared to the *naive* method this method is able to smooth out a certain noise level by averaging over a sliding window. The computational overhead is negligible.

2.3.2.3 Simple Exponential Smoothing

The *simple exponential smoothing* (SES) method extends the moving averages approach by weighting more recent values in a sliding window with exponential higher factors. In the first step, the parameters of this exponential function are adapted to the given time series data in an iterative optimisation process beginning with recommended start parameters, before in the second step, point forecasts and confidence intervals are computed iteratively. This method smooths out a certain noise level and reacts more flexible as the *arithmetic mean* method on the influences of trend or seasonal patterns due to the exponential weighting. But still, as this method inherently damps any changes in the values it does not extrapolate trends or other developments due to seasonal patterns. Experiments showed that the SES method returns a result below 80milliseconds when applied on less than 100 values. This computational overhead is mainly induced by the parameter estimation step. SES is suitable for short-term forecasts and can be applied already at a small time series size. In the SES method no trend or season component is considered, but it is extended to do so in the ETS method. The SES method is equal to an application of the ARIMA$((p, d, q) = (0, 1, 1))$ model as described in 2.3.2.9.

2.3.2.4 Cubic Smoothing Splines

As demonstrated and discussed in [HKPB02], *cubic smoothing splines* (CS) can be fitted to univariate time series data to obtain a linear forecast function that estimates the trend. The smoothing parameters are estimated using a likelihood approach enabling the construction of confidence intervals. The authors say that this approach is a special case of the ARIMA$((p, d, q) = (0, 2, 2))$ model as described in 2.3.2.9 with a restricted parameter set. They demonstrate that this restriction does not impair the forecast accuracy. This approach tends to better estimate trends than the SES method, but seasonal patterns cannot be captured. This method sometimes overestimates a trend in steep parts of a time series. The computational overhead stays below 100 milliseconds when applied on the last 30 values. It has been observed that the computation time rises for more values without an observable improvement in forecast accuracy.

2.3.2.5 ARIMA(1,0,1) Stochastic Process Model

The ARIMA$((p, d, q) = (1, 0, 1))$ model is an instance of the ARIMA (auto-regressive integrated moving averages) framework as described in Section 2.3.2.9 and assumes a stationary stochastic process (constant mean value) as it does not make use of integration

as in SES and CS, where d is not zero in the corresponding ARIMA models. Therefore, it can also be called an ARMA($(p, q) = (1, 1)$) process model as explained in Shumway's book on time series analysis [Shu11]. This method is not as sensitive as the CS method to steep parts in a time series. This application of the auto-regressive moving average includes as in SES and CS a parameter estimation that induces the major computational effort. Experiments showed that this method returns results below 70 milliseconds when applied on the last 100 values.

2.3.2.6 Croston's Method for Intermittent Time Series

Croston's method is presented in [SH05] and is specialized for forecasting of intermittent time series. In contrast to the other methods, this is applicable to time series that contain zero values. Internally, the original time series is decomposed into a time series without zero values and a second one that captures durations of zero valued intervals. These two time series are then independently forecast using the SES method and unified. As this method has no consistent underlying stochastic model, confidence intervals cannot be computed. As this method is based on the SES method, the computational overhead is slightly higher with 100 milliseconds computation time on 100 values.

2.3.2.7 Extended Exponential Smoothing

As presented in [Hyn08] and [HK08], the *extended exponential smoothing* (ETS) bases on the innovation state space approach and explicitly models a trend, a season and a trend component in individual SES equations that are combined in the final forecast result in either additive or multiplicative (or neglected) manner. In addition, damping the influence of one of these components is possible. According to this model framework, the forecast process starts with the selection of an optimized model, before the parameters of the single SES equations are estimated. Having the model and the parameters adapted to the time series data, point forecasts and confidence intervals are computed. This method is able to detect and capture sinus like seasonal patterns that are contained at least three times in the time series data. In this case, the ETS has a computation time of 15 seconds on 200 time series values. In the case of more complex patterns, this method is a multiple faster, but not able to detect this pattern resulting in worse forecast accuracy.

2.3.2.8 tBATS Innovation State Space Modelling Framework

The *tBATS* innovation state space modelling framework has been presented in 2011 in [DLHS11] and recently been integrated into the R forecasting package. It further extends the ETS state space model for a better handling of more complex seasonal effects by making use of a trigonometric representation of seasonal components based on a Fourier transformations, by the incorporation of Box-Cox transformations and by use of ARMA error correction. tBATS relies on a new method that reduces the computational burden of the maximum likelihood estimation. In experiments processing times of up to 18 seconds have been observed on 200 values, but in several cases the processing time of five to seven seconds still resulted in appropriate forecast accuracy.

2.3.2.9 ARIMA Stochastic Process Modelling Framework

The *ARIMA* (auto-regressive integrated moving averages) stochastic process modelling framework is presented in the book "Time Series Analysis - Forecasting and Control" by Box, Jenkins and Reinsel [BJR08]. The ARIMA model space is defined by seven parameters (p, d, q) and $(P, D, Q)_m$, whereas the first triple defines the model concerning trend and noise component, the second vector is optional and defines a model for the seasonal component. The parameter m stands for the frequency of the seasonality. P or

p stands for the order of the AR(p) process, D or d for the order of integration (needed for the transformation into a stationary stochastic process) and Q or q for the order of the MA(q) process. This model space is theoretically unlimited as the parameters are positive integers or zero. The model selection is a difficult process that can be realised via space limitation and intelligent model space traversion using different unit-root tests (KPSS, HEGY or Canova-Hansen) and Akiake's information criterion (AIC). Hyndman proposes an process for automated model selection in [HK08] that is implemented in the `auto.arima()` function of the R forecast package. A selected ARIMA model is then fitted to the time series data to compute point forecasts and confidence intervals. A high computational effort is induced by the model selection and further fitting: Up to 50 seconds on 200 values, but with a high variance as the model selection process depends also on the data itself and not only on the amount of data to base the forecast result on. Experiments also showed that this ARIMA approach achieves in most cases closer confidence intervals than the tBATS approach.

2.3.3 Forecast Accuracy Metrics as Feedback for Classification

Numerous error metrics have been proposed to capture the differences between point forecasts and corresponding observations and are summarized, explained and compared in the publication of Hyndman and Koehler [HK06]. In the following, the forecast accuracy metrics are defined and grouped into four sets as presented in the mentioned publication.

Table 2.3: Scale-dependent Error Metrics

Mean Square Error	MSE	$mean({e_t}^2)$
Root Mean Square Error	RMSE	$\sqrt{MSE}$
Mean Absolute Error	MAE	$mean(\|e_t\|)$
Median Absolute Error	MdAR	$median(\|e_t\|)$

The first group of error metrics in Table 2.3 are the scale-dependent ones that are directly computed using the error e_t at a time point t: $e_t = forecastValue_t - observedValue_t$ and can be used for comparisons of forecast accuracy on identical data.

Table 2.4: Percentage Error Metrics (Scale-independent)

Mean Absolute Percentage Error	MAPE	$mean(\|p_t\|)$
Median Absolute Percentage Error	MdAPE	$median(\|p_t\|)$
Root Mean Square Percentage Error	RMSPE	$\sqrt{mean({p_t}^2)}$
Root Median Square Percentage Error	RMdSPE	$\sqrt{median({p_t}^2)}$

The second group are the percentage error metrics in Table 2.4 that relate the errors e_t to the corresponding observed value at the time point t: $p_t = \frac{100 \times e_t}{observedValue_t}$ These metrics are undefined, if the time series contains zero values.

The third group contains relative error metrics as in Table 2.5 that relate the errors e_t of the method in focus to the errors ${e_t}^*$ of a benchmark method: $r_t = \frac{e_t}{{e_t}^*}$. The benchmark method need to be defined beforehand to enable accuracy comparisons of forecast methods even on different data samples.

The fourth group are the scaled error metrics as in Table 2.6 that relate the errors e_t to the in-sample forecast results of a one-step naive forecast as follows:

Table 2.5: Relative Error Metrics (Scale-independent)

Mean Relative Absolute Error (MRAE)	$mean(\|r_t\|)$
Median Relative Absolute Error (MdRAE)	$median(\|r_t\|)$
Geometric Mean Relative Absolute Error (GMRAE)	$\sqrt{gmean(\|r_t\|)}$

Table 2.6: Scaled Error Metrics

Mean Absolute Scaled Error (MASE)	$mean(\|q_t\|)$
Median Absolute Scaled Error (MdASE)	$median(\|q_t\|)$
Root Mean Squared Scaled Error (RMSSE)	$\sqrt{mean(q_t)^2}$

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |observedValue_i - observedValue_{i-1}|}$$

This way, the scaled error are relative errors that use the naive forecast method as benchmark. The metric is easily interpretable as a value smaller than 1 indicates smaller baseline than compared to errors of the naive method and a larger value indicates a worse accuracy accordingly.

In [HK06], the authors propose to use the MASE forecast accuracy metric to enable consistent comparisons of forecast strategies even over different data samples. Computing the MASE metrics enables direct impressions whether the forecast result is better as the naive approach and therefore indicates that a result interpretation may be useful for proactive provisioning. The R forecast package offers the `accuracy()` method that offers inter alia the MASE metric that is based on the recently available time series data and captures the historic forecast accuracy as an estimate for the recent forecast.

2.4 Application Context of WIB Forecasts

Having continuous forecasts with an appropriate accuracy that are calculated online in a repetitive manner and induce controllable processing overheads, enables to estimate a future usage profile of a software service and is therefore an building block for the realisation of a proactive resource management. The plausibility of the forecast results can be directly evaluated by such an resource management using the provided estimates for the recent accuracy in form of confidence intervals and the MASE metrics. An intelligent system, in which these approaches are integrated, can be aware of its near future and manage its resources proactively in addition to reactive mechanisms.

The system for workload classification and forecasting as introduced in the following chapters of this thesis can be integrated into an comprehensive process for intelligent reactive and proactive resource provisioning as is illustrated in the chart in Figure 2.2 from [KBHR10]. The incorporation of a workload forecasting system is visualised as the box in the upper right side to provide continuous forecast results of monitored arrivals rates to a resource management component based on architecture-level performance models. This component takes the predicted usage profile (or workload intensity behavior) as input parameter to estimate resource consumptions. After completion of this step, likely resource shortages or conflicts are detected and solved if possible by finding an optimized reconfiguration scenario.

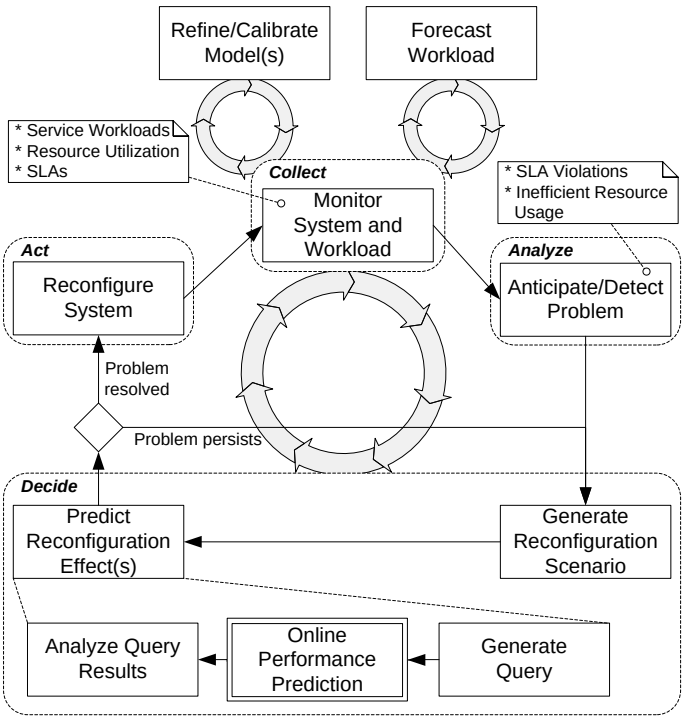


Figure 2.2: Illustration of an Online System Reconfiguration Process as published in [KBHR10]

A nearly optimal resource efficiency may be achieved by dynamic proactive provisioning, when in all points of time the demand for resources is almost equal to the amount of provisioned resource. Any improvement in resource efficiency results in savings of energy and costs. Providing a solution to this dynamic optimisation problem would address the challenge of Software-as-a-Service (SaaS) cloud providers to run their deployed services in an elastic system compliant to performance goals as defined in fixed service level agreements (SLAs), while not wasting resources and energy on the other hand.

3. Approach

As stated in the Introduction in Section 1.1, the major research goal of this diploma thesis is the construction of an automatically executable classification scheme for workload intensity behaviors. An approach for this goal is presented in this chapter that uses a decision tree and forecast quality feedback mechanisms for the computation of classification results to enable the selection of a time series forecast strategy according to a given set of forecast objectives. The classification process and the forecast strategies are both applied periodically. To achieve a trustworthy classification of the workload intensity behavior, a subset of suitable forecast strategies is selected according to given objectives and properties of the time series. After execution of different forecast strategies in this subset, their accuracy is evaluated for the classification. By taking forecast quality metrics as feedback into consideration, the overall forecast accuracy is improved in the majority of cases as shown in the Evaluation in Chapter 5 of this thesis.

In Section 3.1 assumptions and inevitable limitations of the approach are outlined and discussed. In the second part of this chapter 3.2 the basic ideas and concepts of the approach are presented stepwise and detailed. The composition of these concepts within a unified classification process can be found in Section 3.3.

3.1 Assumptions and Limitations

It is inevitable to assume that a software service, on which this workload classification and forecasting (WCF) approach is applied, is able to constantly monitor and report its request arrivals of a defined request class, for example by using a monitoring framework like *Kieker* which is presented in [vHWH12]. In this case time series based forecast algorithms can be applied to predict future time series points. Concerning this approach, the input to a time series based forecast algorithm is assumed to be a time series of request arrival rates of a single request class. This implies that the focus is not to observe or analyze interactions between different two or more different request classes, which would be the task of system resource planning and optimization algorithms like presented in a broad spectrum of research papers [MIK+10, vMvHH11, KBHR10, KKK10, JHJ+10, HBK11, HMBN10, CEM+10, ACCM09, BKB07].

In addition, this WCF approach does not analyse request characteristics or predict the impact of changes within the arriving requests themselves like changes in data sizes or complexity as discussed for example in [AW96, CS85, Raa93]. Even though the request characteristics are essential for the estimation of the resource demand. A trustworthy resource

demand estimation can be achieved by means of system monitoring and is needed for calibration of a system's performance model that is able to simulate and predict performance metrics for a given arrival rate of requests. Approaches for resource demand estimation and performance model calibration are presented in [Kou07, KBHR10, Kou05, HBK11].

Moreover, this WCF approach is not meant to be applied to time series containing response time, resource utilization or throughput values. Forecasting of these quality of service attributes making use of time series analysis methods is in the focus of approaches that are shortly discussed in the context of related work in Chapter 6. The basic performance metrics characterize a system's capability to handle and process requests but do not directly quantify the amount of arriving work. In the wider context of this thesis, arrival rates are not a performance metric but serve as input for performance models, which are later analysed. Due to the fact that the stochastic models of the forecast strategies based on time series analysis are not designed and therefore not capable to capture all system's performance relevant characteristics in addition to the workload intensity behavior. Therefore, a direct prediction of performance metrics may be difficult to achieve with a high accuracy needed for trustworthy decisions.

Any forecast is inherently connected with uncertainty due to unforeseeable system extrinsic influences. It is obvious that the influence of a system extrinsic change or event as for example an accident cannot be forecast based on experience that is captured solely in system specific historic data. This is due to the non-deterministic nature of apparently randomized events and their possibly sudden influence to a system. In contrast, the influence of a planned change or event could be manually estimated beforehand. Unplanned events can cause anomalies that can be detected and analyzed afterwards by methods like ΘPAD. Speaking in basic terms, this method compares the observation with the expected values. Performance anomaly detection is the target of a related diploma thesis at Christian-Albrecht University Kiel authored by Tillmann Carlos Bielefeld [Bie12].

The uncertainty induced by sudden anomalies and constant noise is usually captured by confidence intervals of a given confidence level α. These intervals symmetrically enclose the forecast mean value. Deterministically recurring fluctuations and patterns form the seasonal component within the workload intensity behavior and need to be observed at least two or better three times to enable their detection and analysis of their temporal distance and shape by the more complex forecast strategies like ARIMA or tBATS.

3.2 Concepts for the Classification of a Workload Intensity Behavior

This section presents and discusses step by step the basic concepts for the classification of workload intensity behavior that are then combined into a unified classification process via decision tree with feedback cycles as shown in Section 3.3

3.2.1 Workload Intensity Behavior Classes

A workload intensity behavior class is defined via the current choice of the most appropriate forecast strategy. Each class identifier is equal to the name of the corresponding forecast strategy. A workload intensity behavior changes and develops over time in a way that the classification is not stationary and therefore needs periodically verification with may lead to a reclassification. For example, a workload intensity behavior at a time point t is in the class that is named after the currently selected forecast strategy A delivering optimized forecast accuracy compared to other available forecast strategies. At a later point in time $t+i$ the workload intensity behavior has been reclassified and its class is now named after a forecast strategy B.

As different forecast strategies possibly deliver identical results due to an overlap of their underlying stochastic model, the class of a workload intensity behavior may not be unique anytime.

3.2.2 Time Series Attributes

A time series of request arrival rates of a request class is a dynamic and frequently updated object and contains the most recent available request arrival rates within the single time series points up to a defined length of the time series. For this WCF approach, the length of a time series should be limited to a maximum of 200 values to assure that the processing times of a single forecast strategy execution stays below 60 seconds. Due to high algorithmic complexity classes of the more advanced forecast strategies the processing times grow quickly with a higher amount of time series values.

According to the definition of a time series in Section 2.1 a time series object has five attributes, that are fixed for the lifetime of a time series instance and further explained in Table 3.1.

Table 3.1: Time Series Attributes

Parameter Name	Parameter Range	Proposed Configuration	Explanation
StartTime	[0;max_long][ms]	given by monitoring setting	The timestamp (absolute time in [ms]) of the first arrival rate value added to the time series is equal to the start time of the time series
DeltaTime	(0;max_long]	given by monitoring setting	The constant time difference between two time series values is the *Delta Time*
DeltaTime Unit	{ms; sec; min; hours; days; weeks; months; years}	given by monitoring setting	This parameter defines the time unit of the *DeltaTime* parameter
Frequency	[1;max_int]	[7;65]	The *Frequency* is the number of time series points that add up either to the next bigger time unit and/or to the estimated length of seasonal patterns in focus. The value should not be too small (still able to approximate the shape of the seasonal pattern) and not to high (to limit the computational effort of complex forecast strategies)
Maximum Periods	[1;max_int]	[3;28]	The amount of *Frequency* time series points form a period. This parameter defines the maximum number of periods that fit into the time series. As in a `fifo' queue the oldest values fall off when more recent values are added. The value of this setting should be at least 3 to enable reliable pattern detection by complex forecast strategies and multiplied the by *Frequency* value not be higher than 200 if the computational effort of more complex forecast strategies should stay below one minute.

3.2.3 Overhead Groups of Forecast Strategies

The in 2.2 presented forecast strategies are grouped into four subsets according to their computational overhead:

- *Group* 1 stands for nearly none overhead and contains *mean value forecasting* and *naive forecasting*.
- *Group* 2 stands for low overhead and contains the fast forecasting strategies *Simple Exponential Smoothing (SES)*, *Cubic Spline Interpolation (CS)*, the predefined $ARIMA((p,d,q) = (1,0,1))$ model and the specialized *Croston's method*. The processing times of of forecast strategies in this group are below $100ms$ for a maximum of 200 time series points.
- *Group* 3 stands for medium overheads and contains the forecasts strategies *Extended Exponential Smoothing (ETS)* and the *tBATS approach*. The processing times are below 30 seconds for less than 200 time series points.
- *Group* 4 stands for high overheads and contains again the *tBATS approach* and additionally the *ARIMA forecasting* framework with automatic selection of an optimal

ARIMA model. The processing times stay below 60 seconds for less than 200 time series points.

The four overhead groups of forecast strategies are presented in 3.2 together with their application scenario.

Table 3.2: Overhead Groups of Forecast Strategies

Overhead Group	Strategies	Application
1 – nearly none	Naïve, MovingAverages	These two strategies are only applied if less than *InitialSizeThreshold* values are in the time series. The arithmetic mean strategy can have a forecast accuracy below 1 and therefore be better than a solely reactive approach using implicitly the naïve strategy. This is only true in cases of nearly constant base level of the arrivals rates. These strategies should be executed as frequently as possible every new time series point.
2 - low	CubicSmoothingSplines, ARIMA101, SimpleExponential Smoothing, Croston's method for intermittent demands	The strengths of these strategies are the low computational efforts below 100ms and their ability to extrapolate the trend component. They differ in sensitivity to noise level or seasonal components. These strategies need to be executed in a high frequency with small horizons.
3 - medium	ExtendedExponential Smoothing, tBATS	The computational effort for both strategies is below 30 sec for a maximum of 200 time series points. They differ in the capabilities of modeling seasonal components.
4 - high	ARIMA, tBATS	The computational effort for the ARIMA approach can reach up to 60 sec for a maximum of 200 time series points and may achieve smaller confidence intervals than the tBATS approach.

3.2.4 Forecast Objectives

As the forecast results can be used for a variety of purposes like manual long term resource planning or short term proactive resource provisioning, a set of forecast objectives is introduced to enable tailored forecast result processing and overhead control. The following parameters should be configurable in a forecast objectives object:

- The *Highest Overhead Group* parameter should be a value of the interval $[1; 4]$ and defines the highest overhead group from which the forecast strategies are allowed to be chosen in the classification process.
- The *Forecast Horizon* parameter is given by a tuple of two positive integer values quantifying the number of time series points to forecast. The first value defines the start value, the second value accordingly the maximum forecast horizon setting. Due to significant differences of the forecast strategies in processing times and capabilities for long term forecasts, the start value is increased by multipliers that are defined in the configuration of the classification process up to the given maximum value.
- The *Confidence Level* parameter can be a value out of the interval $[0; 100)$ and sets the confidence level *alpha* in percent for the confidence intervals surrounding the forecast mean values.
- *Forecast Period* is an positive integer parameter i. Every next i new time series points in the time series of request arrival rates a forecast will be triggered.

The configuration of these forecast objectives allows a customization of the execution of the forecasts and their results. Proposed settings and further explanations can be found in Table 3.3.

Table 3.3: Forecast Objective Attributes

Parameter Name	Parameter Range	Proposed Configuration	Explanation
ForecastPeriod	[1;max_int]	[1; *Frequency*]	This objective defines how often a forecast is executed in times of new time series points. For a value of 1 a forecast is requested every new time series point and can be dynamically increased by period factors in the classification setting to reach the configured maximum horizon. This value should be equal or smaller than the *StartHorizon* objective (if continuous or even overlapping forecasts are needed)
Highest Overhead Group	[1;4]	[2;4]	This objective defines the highest overhead group from which the forecast strategies will be chosen. A value of 2 may be sufficient if the time series data have strong trend components that are not overlaid by seasonal patterns, as the strength of group 2 strategies is the trend extrapolation. For time series with seasonal patterns, a setting of 3 for a maximum forecast computation time of 30 seconds and 4 for forecast computation times below 1 minute is recommended.
ConfidenceLevel	[0;100)	may be given by a forecast interpreter	The confidence level α of the returned forecast confidence intervals is defined by this objective.
StartHorizon	[1;max_int]	[1; 1/8x *Frequency*]	The S*tartHorizon* defines the number of time series points to be forecasted at the beginning and can be dynamically increased by period factors in the classification setting up to the *MaximumHorizon* setting. This value should be equal or higher than the *ForecastPeriod* objective (if continuous or even overlapping forecasts are needed).
Maximum Horizon	[1;max_int]	*Frequency*	The value of *MaximumHorizon* setting defines the maximum number of time series points to be forecasted. A recommendation for this setting is the value of the *Frequency* setting of the time series, as a higher horizon setting may lead to broad confidence intervals.

3.2.5 Partitions of the Classification Process

According to the theory of neuronal stimuli processing by Joseph LeDoux, stimuli are processed by the human brain in two different and independent ways to fulfill different objectives. The first is needed for a fast but possibly inaccurate result, the second for a slower but far more precise and reliable result.

Motivated by this theory from the field of human biology, the classification process of workload intensity behaviors can be partitioned accordingly. When a time series of request arrival rates is newly registered at the classification system and no historic data on the workload intensity behavior is available yet, it is solely useful to apply fast and naive strategies just to get a basic forecast. At a later point in time when some observations are already available, it may be useful to apply strategies that can interpolate the trends within the time series. And again at a later point in time, when about three periods within the time series have already been observed, it is possible to detect deterministic seasonal patterns by using complex time series analysis, decomposition and forecasting methods.

The classification process is designed to have an initial, a fast and a complex partition according to the amount of available historic data in the time series. Having a short time series only forecast strategies of the *overhead group 1* can be applied. A medium length may allow application of strategies contained in the *overhead group 2* and a long time series enables the use of the strategies in *overhead group 3 and 4*. The two thresholds that define, when a time series is short, medium or long can be set as parameters in the classification setting.

Based on experience gained from experiments it is my recommendation to set the low-medium threshold to a value between five (as this is the minimal amount of point needed for Cubic Spline Interpolation) and the value that is half of a period. The medium-long threshold should be set to a value as high as three time series periods for the simple reason that the most methods for the recognition of seasonal patterns need at least three occurrences.

A short summary of the capabilities of this three different classification strategies (initial, fast and complex) can be found in Table 3.4. How these capabilities are realised, is discussed in detail in the following sections.

Table 3.4: Classification Strategies and their Capabilities

Classification Strategy	ForecastStrategy OverheadGroup	Capabilities
Initial	1 – nearly none	This classification strategy only checks via the estimated and observed MASE metrics whether the arithmetic mean is a better forecast estimate than the naïve approach.
Fast	2 - very low	This classification strategy observes the noise level and can apply the moving averages for smoothing, heuristically selects the *Croston's* method or the cubic spline interpolation, evaluates the result plausibility and the forecast accuracy via the estimated and observed MASE metrics to adjust the current classification.
Complex	3 – medium, 4 - high	This classification strategy observes the noise level and can apply the moving averages for smoothing, heuristically selects the *Croston's* method if necessary and evaluates the result plausibility and the forecast accuracy via the estimated and observed MASE metrics to adjust the current classification. Either strategies from overhead class 3 or class 4 are selected.

3.2.6 Handling of Zero Values

A time series of request arrival rates contains positive integer values and also zero values, as a system may no be in constant use. The majority of forecasting strategies is possibly not numerical stable if the input time series contains zero values. This would mean that a forecast strategy interrupts after a division by zero exception and cannot return a forecast result. This fact makes a simple elimination of zeroes before passing to the forecast strategy necessary.

If the zero values in the time series are not only a single event but arise regularly, it is better to use the special Croston's forecasting strategy for intermittent demands that decomposes the time series into two different ones: A strictly positive time series and another time series containing the period duration as points when zero values are observed. The forecast is then executed independently for both time series and combined later on.

The classification of workload intensity behavior would be highly sensitive to zero values, if only a few observations of a zero values would immediately trigger a switch to the Croston's forecast strategy. To make this sensitivity configurable a threshold for the rate of zero values is introduced and recommended to be set to a reasonable value between 20% and 40%.

3.2.7 Forecast Quality Improvement using Weighted Moving Averages

A high noise level within the time series of request arrival rates complicates every forecasting process. As the workload intensity behavior is not stationary, the level of present noise even may change over time. To improve the quality of forecasts it is conducive to observe metrics that capture the noise level within the time series and define thresholds for these metrics. These threshold can trigger a smoothing method before the time series is processed by the forecasting strategy. However, any smoothing method has to be applied carefully not to eliminate valuable information.

The following four metrics that were already introduced in Section 2.2 and in Table 2.1 are suitable and sufficient to quantify noise related characteristics within a time series and therefore should be repeatedly applied to a configurable amount n of the most recent time series points:

- *Burstiness Index:* A burstiness index close to zero indicates strong bursts within the analysed time span.
- *Relative Length of the Longest Monotonic Section:* If the longest monotonic section related to the length of the observed time series span is close to zero, this metric

indicates frequent fluctuations. The application of a smoothing method would clarify the trend and seasonal pattern.

- *Variance Coefficient:* The variance coefficient is a common, unit-free measure of variance. If the value of this coefficient is one or bigger, the standard deviation is higher than the mean value. A smoothing method can slightly reduce the level of the variance, leading to a clearer visibility of trends within the time series.
- *Quartile Dispersion Coefficient:* The quartile dispersion coefficient is a unit-free metric to quantify the range size of the time series values as well as the amount of outlying values whose negative effect on the forecast could be minimized by a smoothing method.

These metrics are easy to compute without high processing overheads. But as none of these metrics directly measures the noise level, Based on experiments it can be recommended that at least three of these four metrics should exceed the configured thresholds, before a smoothing technique should be applied.

Smoothing of the time series can be achieved for example by the elementary method Weighted Moving Averages (WMA), where the level of smoothing can easily be controlled by the window size and the weights. For this WCF approach it is recommended to use a small window size of only three, a weight vector $w = (0.25; 0.5; 0.25)$ and a control variable to assure, that no value is smoothed more than one time. This setting would slightly smooth out a high noise level but avoids the loss of valuable information.

Time series smoothing can also be achieved by far more complex methods like using a Fast Fourier Transformation to build a low-pass filter eliminating high frequencies. However, this would introduce additional overheads and would require an adequately high amount of time series data.

3.2.8 MASE Metric for Decisions between Alternatives

As already mentioned, the workload intensity behavior is supposed to be non-stationary - in other words, its class and accordingly the most suitable forecast strategy will change from time to time. It is the responsibility of the classification process to identify the current workload intensity behavior class that is needed for the selection of the most suitable forecast strategy according to the given forecast objectives.

The evaluation and comparison of the forecast accuracy gives crucial input for this decision making process. If the strategy delivers results of low accuracy or even non-plausible results, the strategy need to be compared directly to one or more other forecast strategies. In this case, two or even more forecast strategies will be executed in parallel on the same time series data. Before the following forecast on the next period will be triggered, the performance of the parallel executed can be evaluated just by a comparison of the forecast values to the observed ones. The strategy that shows the highest accuracy defines the workload intensity behavior class. In addition, this mechanism is applied periodically to assure that the classification is still valid.

However, any forecast result can only be of use to a proactive resource planning before the corresponding monitored arrival rates are observed and reported. This fact makes it necessary to additionally estimate the better performing strategy beforehand to select the one that is more likely to show higher accuracy if more than one forecast is computed.

As presented in Section 2.3.3 and extensively discussed in [HK06] the *Mean Absolute Scaled Error* (MASE) metric is the metric of choice to measure forecast accuracy. As the metric directly compares the forecast accuracy to the accuracy of the naive forecast strategy, it is directly visible when a strategy performs poorly. If the MASE is close to one or even bigger,

the computed forecast results are of no value, because their accuracy is even or worse to the naive forecast that is based on the simple assumption that the last observation is still valid for the next period. The closer the metric value is to zero, the better the forecast strategy can deliver trustworthy forecast values.

To implement the above outlined decision making process, the MASE metric is computed at two different times: First, the MASE is calculated at the time point of the forecast itself, capturing the forecast performance in the past to get an estimation for the future time steps. This estimated forecast accuracy is used to decide, which of two or more strategies executed in parallel to be the better one. Second, before the next classification process is triggered, it is now possible to directly measure the forecast performance, as the observed values are already known. This exact MASE metric is now used for the decision making about a need for a reclassification. The forecast strategy having the smallest MASE value defines a possibly changed workload intensity behavior class.

3.2.9 Cubic Spline Interpolation and MASE estimation

Using the Cubic Spline Interpolation method for short term forecasting brings the advantage that this method does not need many time series points to achieve possibly good forecast results with low computational overhead.

As this strategy fits cubic splines to the points of the time series, in other words minimizes the distances between a cubic spline and the time series points, the mean absolute scaled error metric estimating the forecast quality does not deliver useful results for a priori decision making.

This leads to the need of developing a heuristic to decide, when the Cubic Spline Interpolation should be used. The method is sensitive to noise or strong bursts. Therefore, thresholds for the burstiness index and the relative length of the longest monotonic section are introduced. If a low burstiness and long monotone parts have been observed in the most recent n values, the Cubic Spline Interpolation method delivers good trend estimations.

As this method cannot differentiate between a seasonal pattern and a trend, it possibly delivers implausible results in some cases. Especially when applied at the edge of a seasonal pattern, where the discrete function of the time series points is usually very steep, interpolation of this wrong trend is not useful. To prevent this, a third threshold for the relative gradient of the discrete function and a simple plausibility check is build into this approach. The relative gradient indicates whether the time series changed more or less than the median over the last eights of a time series period. A plausibility check may just assure that the forecast values are not negative and not bigger than two times the observed maximum.

In the evaluation section, it is shown, that the Cubic Spline Interpolation method is not the best choice in all cases, but delivers the best accuracy among the overhead group 2 strategies in some special cases. Therefore, this method helps to increase the overall forecast accuracy of the presented flexible WCF approach.

3.3 Decision Tree with Feedback Cycles for Classification

The presented concepts and mechanisms can now be combined to build the flexible WCF approach that automatically adapts to changing workload intensity behavior with the result of optimized and more constantly adequate forecast accuracy. The combination of a classification process with a set of forecast strategies enables a configuration according

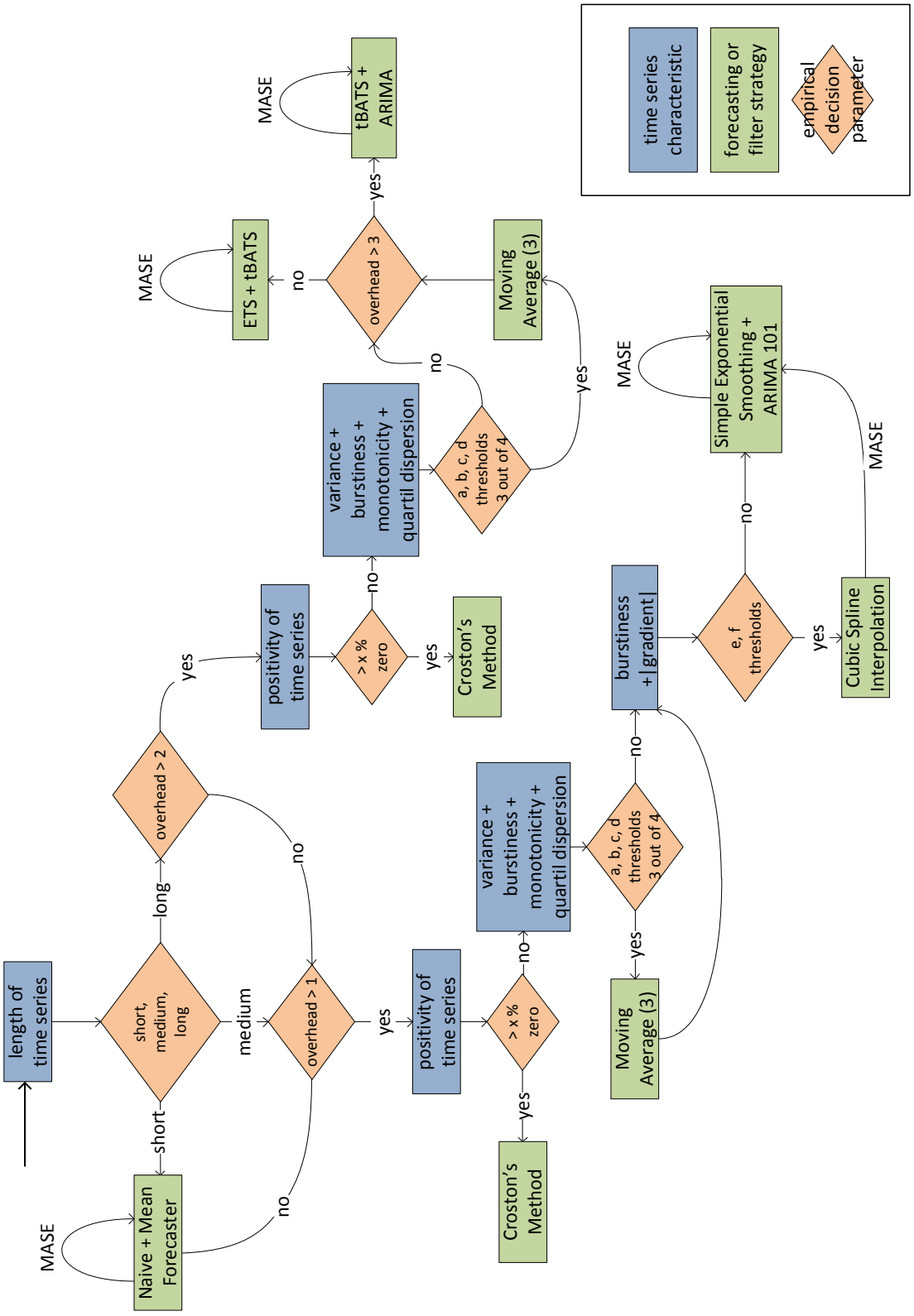


Figure 3.1: Decision Tree for Classification and Selection of a Forecasting Strategy including Direct Feedback Cycles

Table 3.5: Parameters, Parameter Ranges, Further Explanations and Recommended Configurations of the WCF Approach

Parameter Name	Parameter Range	Proposed Configuration	Explanation
Classification Period	[1;max_int]	[1; 4x *Forecast Period*]	This objective defines how often a classification is executed in times of new time series points. For a value of 1 a classification is requested every new time series points and can be dynamically increased by the period factors. A classification can be executed before every single forecast strategy call. It is recommended to set this value equal or smaller than 4 times the *ForecastPeriod* setting (at least every forth forecast call) to detect the time when a reclassification is needed with an adequate precision.
Initial Classification SizeThreshold	[1;max_int]	½ x *Frequency*	This setting defines the time series size at which the classification strategy is switched from initial (overhead group 1) to the fast classification strategy (overhead group 2). The class 2 forecast strategies require at least 4 time series values. Using a rule of thumb this value can be set to half the time series points of a period (as this is in most cases the minimum size for the plausibility check to work properly).
FastClassification Size Threshold	[1;max_int]	3 x *Frequency*	This setting defines the time series size at which the classification strategy is switched from fast (overhead group 2) to the complex classification strategy (overhead groups 3 & 4). The forecast strategies in overhead groups 3 & 4 require at least 3 periods of time series data to detect seasonal patterns.
LastNValues	[1;max_int]	*Frequency*, at least 30	This setting defines the number of the most recent time series points for that the analysis metrics are calculated in the fast or complex classification strategy. The value could be set to the *Frequency* value, to capture the characteristics of the last observed period. The value should be at least as high as 30. As a rule of thumb from common statistics says, the sample variance is an appropriate estimate for the variance of the population for more than 30 values.
FastPeriod Factor	[1;max_int]	[1;4]	This factor setting is a multiplicator for the *StartHorizon* and the *ForecastPeriod* of the forecast objectives as well as for the *ClassificationPeriod* applied when switching from the initial to the fast classification strategy. As the forecast strategies in overhead group 2 only extrapolate the trend components, the *ForecastHorizon* should not growth bigger than an eighth of a period. It may be better to apply forecast strategies in overhead group 2 more frequently, as their computational overheads stay below 100 ms.
ComplexPeriod Factor	[1;max_int]	[1;16]	This factor setting is a multiplicator for the *StartHorizon* and the *ForecastPeriod* of the *ForecastObjectives* as well as for the *ClassificationPeriod* (each already multiplied by the *Fast PeriodFactor)* applied when switching from the fast to the complex classification strategy.
RateZeroValues Threshold	[0;1]	0.3	The rate of zero values defines the sensitivity to zero values in the time series and defines the threshold, when the classification switches to the *Croston's* forecast strategy for intermittent demands. For an observed rate of zero values, these values a just skipped as the other forecast strategies are not numerical stable if the time series contains zero values. As it is not useful to switch the *Croston's* forecast strategy for only a few zero values, it is recommended to set this threshold to a value in [0.2;0.4]
CubicSpline Burstiness Threshold	[0;1]	0.3	This threshold defines a necessary precondition for the heuristic switch to the cubic spline forecasting strategy. The observed burstiness index value of the *LastNValues* need to be higher than the value of 0.3, which can recommend based on experiment experience. The closer the burstiness index is to 0, the more distinctive is the burstiness.
CubicSpline Gradient Threshold	double	0	This threshold defines a necessary precondition for the heuristic switch to the cubic spline forecasting strategy. The observed relative gradient captures whether the discrete function of the time series point changed during the most recent eighths of a period for more (negative value) or less (positive values) than the median value of the *LastNValues*. Based on experiment experience it can be recommended to have this precondition fulfilled for any positive relative gradient.
CubicSpline Relative Monotonicity Threshold	[0;1]	0.2	This threshold defines a necessary precondition for the heuristic switch to the cubic spline forecasting strategy. The observed relative monotonicity is the length of the longest monotone section during the *LastNValues* related to the value of *LastNValues*. The precondition is fulfilled for a higher observed value. Based on experiment experience a threshold of 0.2 is recommended.
Smoothing Burstiness Threshold	[0;1]	0.15	This threshold defines a precondition for the heuristic application of smoothing averages for noise reduction. The observed burstiness index value of the *LastNValues* need to be smaller than the value of 0.15, which can be recommended based on experiment experience. The closer the burstiness index is to 0, the more distinctive is the burstiness.
Smoothing Quartile Dispersion Threshold	[0;max_int]	2	This threshold defines a precondition for the heuristic application of smoothing averages for noise reduction. The quartile dispersion captures is a unit-less index capturing the variance level of the *LastNValues*. If the distance between the quartiles is at least two times bigger than the median value, the precondition would be true for a recommended threshold value of 2.
Smoothing Variance Coefficient Threshold	[0;max_int]	1	This threshold defines a precondition for the heuristic application of smoothing averages for noise reduction. The quartile dispersion captures is a unit-less index capturing the variance level of the *LastNValues*. If the standard deviation is as high as the arithmetic mean value, the precondition would be true for a recommended threshold value of 1.
Smoothing Relative Monotonicity Threshold	[0;1]	0.15	This threshold defines a precondition for the application of smoothing averages for noise reduction. The observed relative monotonicity is the length of the longest monotone section during the *LastNValues* related to the value of *LastNValues*. The precondition is fulfilled for a lower observed value indicating a high noise level. Based on experiment experience a threshold of 0.15 is recommended.

to given forecast objectives. The WCF approach can be controlled in detail and tuned for case specific optimisation via adaptations of the parameter settings and thresholds.

Figure 3.1 illustrates the classification process of the WCF approach. The classification process can be seen as independent from the forecast executions and is triggered according to the configuration of the *classification period* parameter every n time series points, just before a forecast execution is requested. The classification process uses the forecast objectives given by the user and the workload intensity behavior characteristics to reduce the space of suitable forecast strategies. The suitable forecast strategies are then processed together and their estimated forecast accuracy is evaluated to output the more promising result. Before the next forecast strategy execution, the classification (which is the selection of the better performing strategy) is validated by the observed MASE metric and used until the next classification process execution is triggered.

3.3.1 Classification Settings

In Table 3.5, a survey of the parameters and according parameter ranges for the calibration and configuration of the presented WCF approach can be found and in addition further explanations and recommendations for the configuration.

4. Architecture and Implementation

The focus of this chapter is the architecture and implementation of the presented workload classification and forecasting approach and can serve as a high level documentation of the software artifacts that have been developed in the context of this thesis. For documentation and illustration concerns, an UML component and an UML sequence diagram are presented and described in Section 4.1 followed by UML class diagrams on the implementation of the individual components in Section 4.2. In Section 4.3 an in-depth discussion of design decisions is given including remarks on further extension and variation points. A description of the development and run-time environment as well as information on code used from third party and licensing can be found in Section 4.4.

4.1 Architecture of the Workload Classification and Forecasting System

The `WorkloadClassificationAndForecasting` (WCF) system is constructed according to component-based software architecture principles and can be seen as a composite component. An `RForecastServer` instance as a third party component is required by the WCF system to offer the workload classification and forecasting as a service to users or other systems. The WCF system is composed of one composite component and three basic components whose individual responsibilities are presented in the following. The UML component diagram in Figure 4.1 illustrates the compositions and connections of the individual components.

Management: The `Management` composite component is responsible for the realisation of the user's WCF system management requests that arrive at the `WCFSystemManagement` provides interface of the system as discussed in Subsection 4.1.1 In addition, the component manages the data exchange at the `ArrivalRateInput` and `ForecastResultOutput` i/o-interfaces with the users of the WCF system. The `Management` component initiates the classification and forecasting processes according to user specific or default configurations. Another responsibility is the handling of data structures and configurations that need to be held in memory during run-time, as well as their backup for a smooth restart of the system. The `Management` component is composed of the following three basic components:

WIBManagement: The acronym WIB stands for workload intensity behavior. The `WIBManagement` component handles the user's WCF system management re-

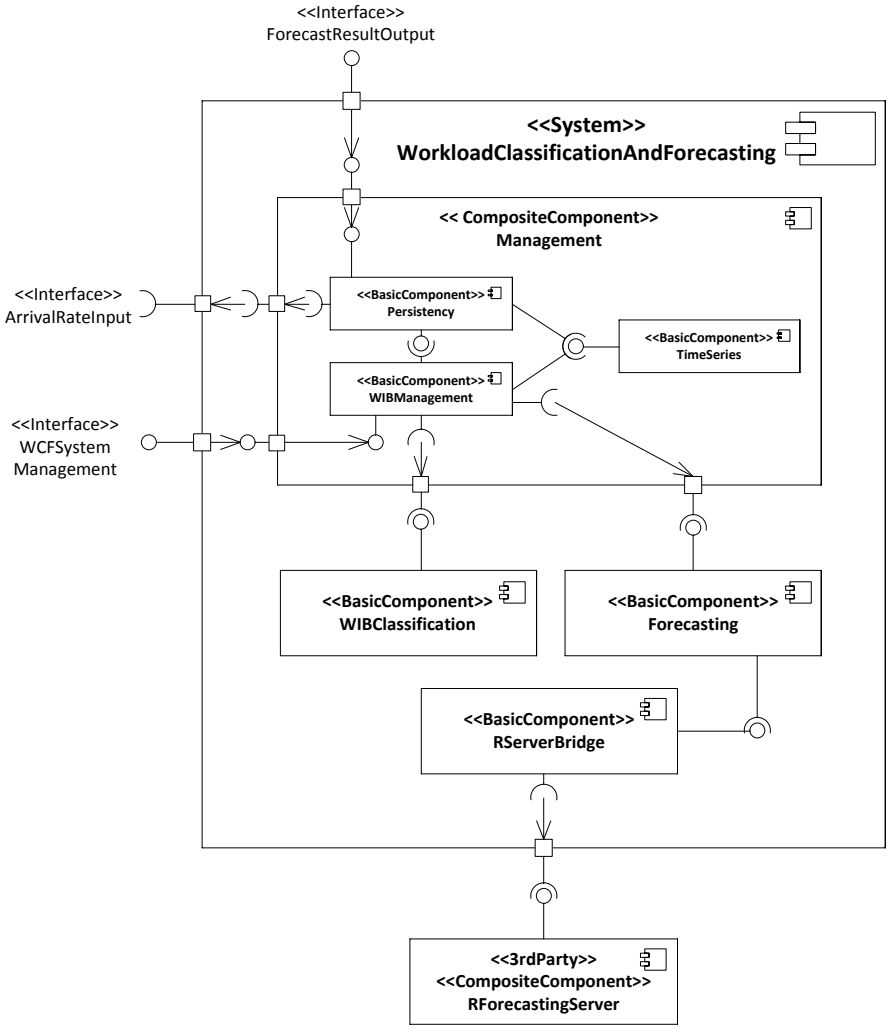


Figure 4.1: UML Component Architecture Diagram of the WCF System

quests and triggers periodically the executions of the classification and forecasting processes according to the user specific or default configurations.

Persistency: The `Persistency` component is responsible for the backup of the of data structures and configurations that are held in memory during run-time, to enable a smooth WCF system restart. In addition, the `Persistency` component offers functionality at the WCF system's `ArrivalRateInput` interface for reading arrival rates and for writing of the forecast results at the `ForecastResultOutput` interface.

TimeSeries: The `TimeSeries` component offers an interface for a time series data structure.

WIBClassification: The `WIBClassification` component is responsible for the classification of workload intensity behaviors according to the presented decision tree in Figure 3.3.

Forecasting: The `Forecasting` component is responsible for the execution of a forecast strategy according to the classification and returns a forecast result.

RServerBridge: The bridge is needed for the communication with the system external `RForecastServer`. This communication includes the composition of scripts and parsing of the result strings in and from the R language for statistical computing [1].

The interactions of these individual components is given in Subsection 4.1.2 and illustrated by an UML sequence diagram in Figure 4.2.

4.1.1 Provides and Requires Interfaces

The `WCFSystemManagement` provides interface offers the management functionality according to the *create-read-update-delete* (CRUD) principle. This enables a WCF system user to register a new or remove a registered `WorkloadIntensityBehavior`, to read and update the `ForecastObjectives` or `ClassificationSettings` of a `WorkloadIntensityBehavior`. The configuration of these two sets of parameters that are presented in Table 3.3 and in Table 3.5 enables the user to define the execution details for periodically classification and forecast processing according to the presented decision tree in Figure 3.1. In addition, the user can manually trigger executions of a classification or a forecast. For WCF system maintenance, a global backup function can be called by a WCF system administrator to allow a system shutdown without loss of run-time data. The WCF system can either be restarted by loading the latest stored configuration using an initialisation routine or started as a clean system by a reset routine.

In addition, the WCF system has a pair of requires `ArrivalRateInput` and provides `ForecastResultOutput` interfaces for the data exchange. For the integration of the WCF system into a monitoring or resource provisioning framework, these two interfaces can be easily adapted as a change impacts only the `Persistency` component. The input data which are newly monitored arrival rate values of a registered `WorkloadIntensityBehavior` can be read for example from a buffer as in common buffered pipes&filters architectures. It is required that the system user provides the monitored arrival rate values constantly in fixed but configurable periods, as the WCF system won't return forecast results in the case that no new values are available. For the evaluation concerns of this approach, the buffer is simulated by a comma separated file (CSV) from which the arrival rate data is read stepwise.

The forecast results could again be written to a buffer if integrated into a pipes&filters architecture to pass it to a resource provisioning system or system adaptation component and allow further processing. For manual result interpretation that is needed for the evaluation of the approach implementation, the forecast results are written to a CSV file.

4.1.2 Exemplary Use Case of the WCF System

The UML sequence diagram in Figure 4.1 illustrates the component interaction for an exemplary use-case of the WCF system showing in addition first implementation specific details of the individual components that are further explained in Sections 4.2.1 to 4.2.3. In the sequence diagram interfaces are used intentionally instead of class instances to underline that the method calls between the components make strict use of provided interfaces.

1. In the first step, the system user registers a workload intensity behavior at the WCF system using the `WCFSystemManagement` interface. The system user can pass customized `ForecastObjectives` and `ClassificationSettings` with this method call, but is not required to provide more than a minimal set of parameters, as the recommended `ClassificationSettings` parameters are chosen by default as described in Table 3.5.

[1] http://www.r-project.org/

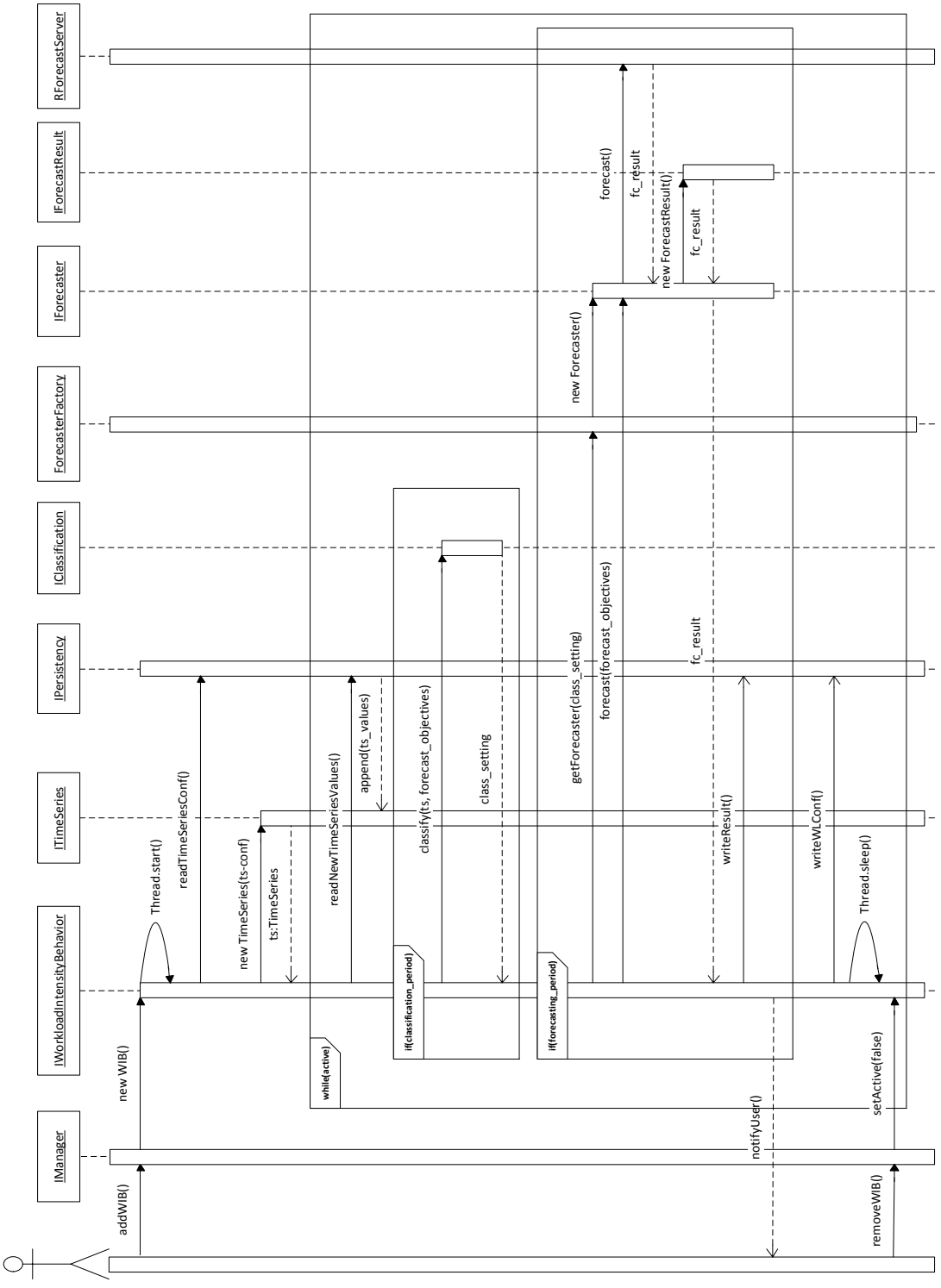


Figure 4.2: UML Sequence Diagram illustrating an Exemplary Use-case of the WCF System

2. This call lets the `Manager` class in the `WIBManagement` component create a new `WorkloadIntensityBehavior` instance as a thread that is started immediately and runs concurrently to other WIB instances. The `WorkloadIntensityBehavior` thread now checks whether a `TimeSeries` for this individual `WorkloadIntensityBehavior` has already been created in the past and if data exist, reads the stored time series configuration to use this for the creation of a new `TimeSeries` in-memory data structure. In the negative case, the user is asked to provide time series configuration parameters as according to Table 3.1 (not modelled).

3. While the `WorkloadIntensityBehavior` thread is not deactivated by its `Manager`, it runs in a loop with a configurable period duration.

   a) In every loop cycle, the interface of the `Persistency` component is called to check for new available arrival rate values that are then appended to the `TimeSeries` in-memory data structure.

   b) In the next step of the loop cycle it is checked, whether a classification is planned, and in the positive case a classification process is synchronously triggered at the interface of the `WIBClassification` component before the thread waits for possible updates to the `ClassificationSetting`.

   c) At this point in the loop cycle it is checked if a forecast is planned to be executed. In the positive case the `ForecasterFactory` of the `Forecasting` component is asked to return a `ForecasterStrategy` object according the result of the classification.

   d) The `ForecasterStrategy` object is now called by the `WorkloadIntensityBehavior` thread to forecast according to the `ForecastObjectives`, which lets the `ForecasterStrategy` object execute an R script on the `RForecastServer` via the `RServerBridge`.

   e) After receiving the forecast results from the `RForecastServer` a new `ForecastResult` object is created, passed backed and printed. If the user listens at an WCF system output buffer, he is automatically notified about new results.

   f) The `WorkloadIntensityBehavior` thread now sleeps for the rest of the configured period duration time.

4. The system user can remove the workload from the WCF system. This does not interrupt a possibly running thread (which could lead to memory leaks) but hinders the `WorkloadIntensityBehavior` thread to reenter its loop.

4.2 Implementation of the Individual WCF System Components

This section presents the implementation and shows UML class diagrams of the individual WCF system components beginning with the `Management` component in Subsection 4.2.1, followed by the `WIBClassification` component in Subsection 4.2.2 and the `Forecasting` component in Subsection 4.2.3.

4.2.1 Management Component

Figure 4.3 shows an UML 2 class diagram of the `Management` component that is composed of the `WIBManagement`, the `Persistency` and the `TimeSeries` components.

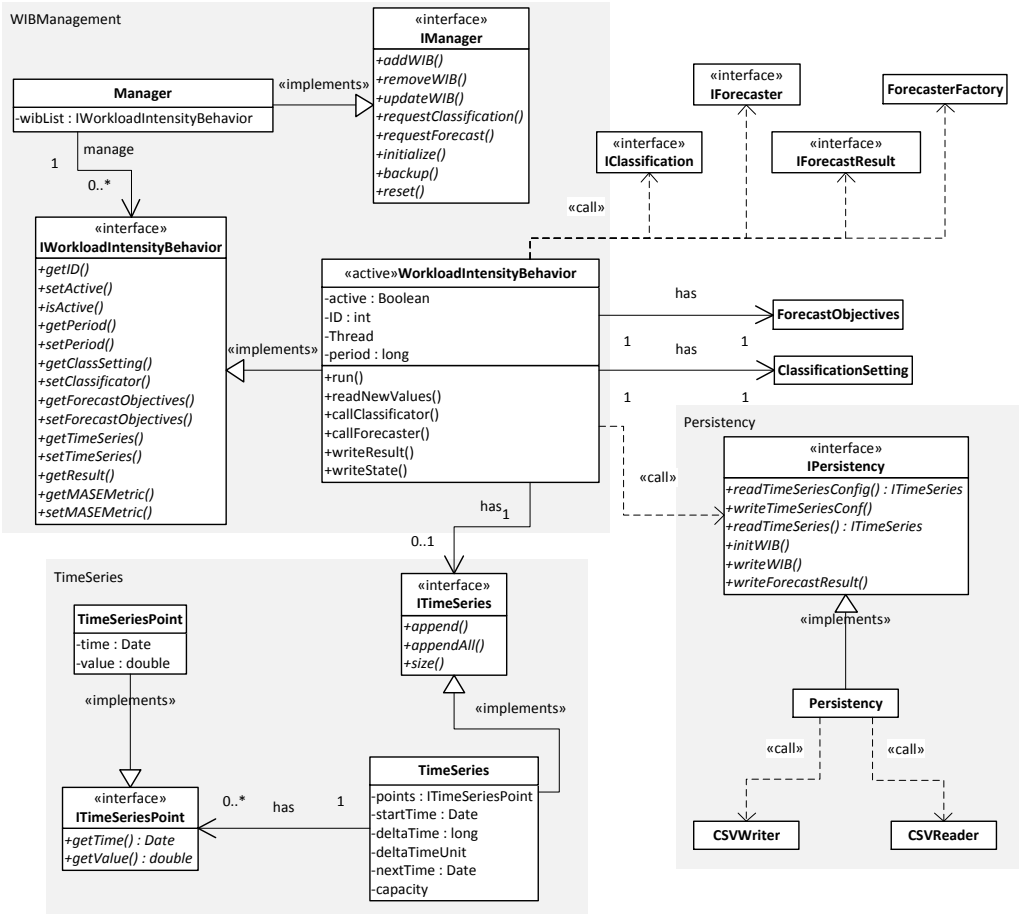


Figure 4.3: UML Class Diagram of the Management Component

4.2.1.1 WIBManagement Component

The `WIBManagement` component is in the upper left quarter of the diagram in Figure 4.3 consisting of the `Manager` class, its interface that is connected the `WCFSystemManagement` user interface and the active `WorkloadIntensityBehavior` instances that are controlled by the `Manager` class via the `WorkloadIntensityBehavior` interface. The `Manager` class has a private list of managed `WorkloadIntensityBehavior` objects.

The active `WorkloadIntensityBehavior` class is realized as a thread and holds references to a `ForecastObjectives` instance of the `Forecasting` component and a `ClassificationSettings` instance of the `WIBClassification` component. These two objects make the classification and forecasting process configurable as outlined in Tables 3.3 and 3.5. In addition, the `WorkloadIntensityBehavior` thread calls the provided interface of the `WIBClassification` component and the static `ForecasterFactory`, the `Forecaster` and `ForecastResult` interfaces of the `Forecasting` component.

4.2.1.2 TimeSeries Component

The active `WorkloadIntensityBehavior` uses the `TimeSeries` component as in-memory data structure which is situated in the lower left of the diagram in Figure 4.3. The `TimeSeries` component itself consists of the `TimeSeries` class that stores the `TimeSeries` configuration attributes in private variables and uses internally the data structure of a circular first-in-first-out (FIFO) buffer (not modelled) to hold the date and value tuples as `TimeSeriesPoints`.

4.2.1.3 Persistency Component

The interface of the `Persistency` component is used by the active `WorkloadIntensityBehavior` class and can be found in the lower right quarter of the diagram. Via this interface, the functionality is offered to read and write a time series configuration, to read new incoming or persisted values, to persist and initialize a `WorkloadIntensityBehavior` configurations and its in-memory data structures and to write out forecast results. The `Persistency` class implements the above mentioned interface and uses in the current code version third party `CSVReader` and `CSVWriter` classes that are publicly available under the GNU licence. This `Persistency` class needs to be adapted if the WCF system should not work directly on the file system but make use of a database or is integrated into a pipes&filters framework.

4.2.2 WIBClassification Component

In Figure 4.4 an UML class diagram of the `WIBClassification` component can be found.

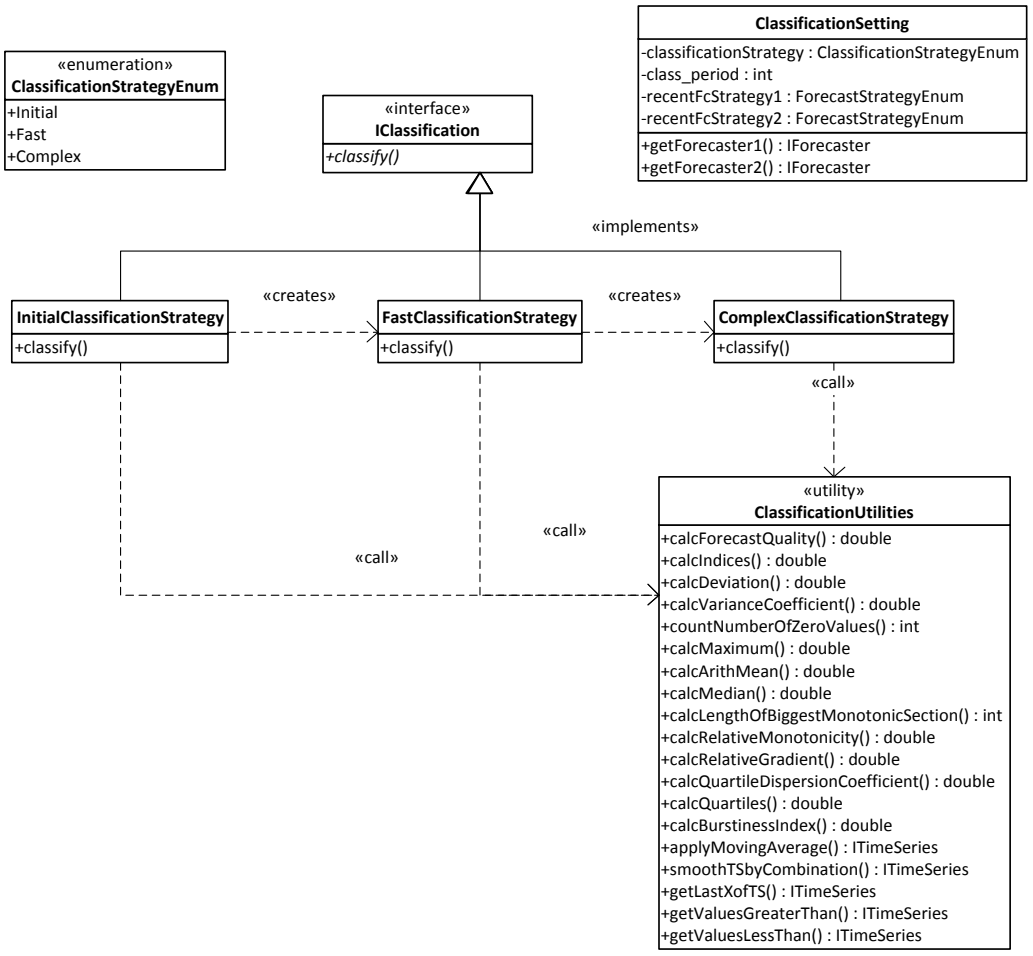


Figure 4.4: UML Class Diagram of the WIBClassification component

The `Classification` interface offers the functionality to the `WIBManagement` component to classify a `WorkloadIntensityBehavior`. The `Classification` interface is implemented by the three concrete strategies `Initial`, `Fast` and `Complex` that are listed in an enumeration. The `ClassificationUtilities` class offers static methods for the computation of the indices and accuracy metrics needed for the classification process as presented in the decision tree in Figure 3.1. The result of a classification, which is a selection of one

concrete `Forecaster` strategies or two strategies if a comparison is needed, is saved and returned in via the `ClassificationSettings` object.

4.2.3 Forecasting Component and RServerBridge

In Figure 4.5, an UML class diagram illustrates the `Forecasting` component implementing the strategy and the factory design patterns. Via the `Forecaster` interface the functionality to initiate a forecast execution is offered to `WIBManagement` component. This interface is implemented by an `AbstractForecaster` that gives the frame for any call to a `ForecastStrategy` and stores the confidence level the configured `ForecastObjectives` and the time series history. The `AbstractForecaster` is further extended by an `AbstractRForecaster` that is R specific and gives a more detailed strategy frame to the concrete `Forecasters` and uses the `RServerBridge` for TCP/IP network communication to the `RForecastServer` instance. The concrete `Forecaster` strategies are created by the `ForecasterFactory` and all of those strategies return a `ForecastResult` object. A `ForecastResult` object contains the forecast duration, the confidence level, the forecast strategy name, a MASE estimation as described in 3.2.8 and finally three `TimeSeries` objects holding the mean forecast value and the upper and lower confidence interval boundaries. All forecast strategies are listed in an enumeration. The `ForecastObjectives` class offers the ability to store a forecast objectives configuration.

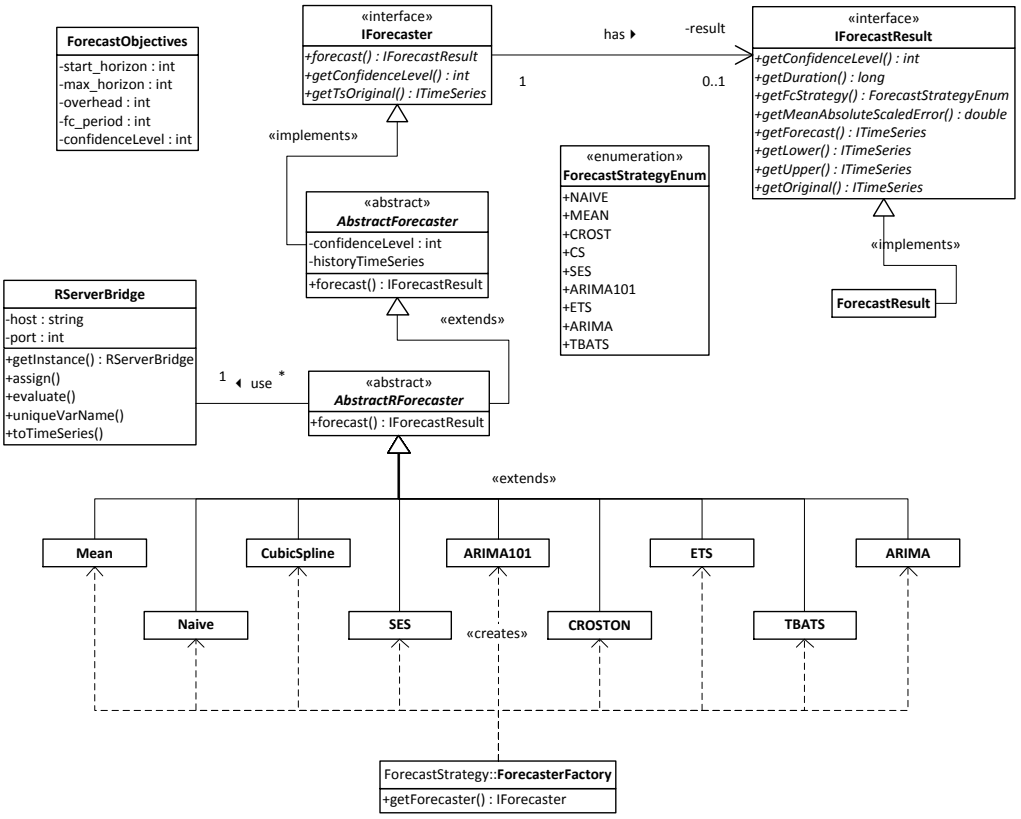


Figure 4.5: UML Class Diagram of the Forecasting Component

4.3 Discussion of Design Decisions

This section discusses several design decisions that have been made during the development of this implementation. These decisions have also been a central topic in an architecture and code review.

Thread Pool Pattern: The use of a thread pool of worker threads controlled by the `Manager` class as an executor with a task waiting queue is seemingly appropriate. But as the workload intensity behavior of a newly deployed and registered request class is meant to be analysed constantly over a longer time period, a thread pool executor distributes mainly long running tasks. The major thread pool benefit of reusing already existing worker threads looses importance in this case. If the manager and not the `WorkloadIntensityBehavior` threads themselves is responsible for the scheduling of single classifications and forecasts, the `Manager` class would grow highly complex as the periods are variable over time and may differ between `WorkloadIntensityBehavior` specific settings. It is difficult to realise a scheduling of classification and forecasting tasks that require to be executed in various periods and additionally need low waiting times in the dispatch queue to guarantee a continuous `ForecastResult` output. Therefore, the design decision has been taken to implement the thread pool pattern with no waiting queue and accordingly an direct dispatch of long running tasks that need no further scheduling. This implies that the `WorkloadIntensityBehavior` threads are responsible for their own individual and configurable scheduling of classification and forecast tasks.

Observer Pattern versus Polling: The use of the observer pattern for notification of the `WorkloadIntensityBehavior` threads for newly available monitoring values has been proposed in the review. In this case, a monitoring system triggers indirectly the classifications and forecasts and is therefore responsible for the exact timing to achieve a continuous `ForecastResult` output. But any monitoring system connected to the WCF system should only be responsible for reliably providing the monitored arrival rates at defined buffers or CSV files. Optionally, a monitoring system is enabled as a WCF system user to manually trigger classifications or forecasts if needed. Therefore, it has been decided not to apply the observer pattern and let the `WorkloadIntensityBehavior` threads periodically check the availability of new arrival rates.

Remote versus Local Forecast Processing: At the beginning of the development it has been a crucial point to decide whether to apply a client-server architecture for the actual forecast result processing or to use local method calls to an R engine. Experiments have shown that the computational overhead for the forecast result processing is high and reaches up to 60 seconds for a single execution using only 200 values. In addition, the amount of data that needs to be transmitted for a forecast execution in a R engine, can be disregarded. Therefore, it was decided to use a client-server architecture to enable local separation the WCF system and the R engine and therefore enable horizontal scaling as the computational resources for the forecast result processing are likely to become a bottleneck resource. The induced network latencies for the communication between the WCF system and the R Server can be kept below an acceptable duration threshold of five milliseconds by using a local area and not a wide area network.

Time Series Data Structure: Having for example an unlimited queue as data structure, it would grow uncontrollable and slow down the WCF system and the forecast processing of the `RForecastServer`. The decision has been taken to use a circular FIFO buffer, which is a simple array of fixed length in a combination with a pointer, as internal data structure of the `TimeSeries` component to handle the `TimeSeriesPoints`. This decision is based on the need to limit the size of the time series to a configurable value. The `TimeSeries` data structure is not thread safe in the current version of the implementation, as no concurrent accesses can happen. Any `TimeSeries` holding historic data is only known to one `WorkloadIntensityBehavior` and any `TimeSeries` in a `ForecastResult` object is created and written once. For use of

the `TimeSeries` data structure in another context with concurrent accesses it would be necessary to synchronize the write and read blocks.

Persistency Layer: For the concern to evaluate the classification and forecasting approach it is sufficient to work directly on the file system using CSV files to simulate input, and store status backup and output. For an integration of the WCF system into a productive software system, the persistence layer can easily be adapted to make use of Java Persistence API or a SQL based database for internal status backups. To improve the communication flow with other systems, the WCF system's persistence layer can be adapted to integrate the WCF system into a pipes&filters architecture.

4.4 Development and Run-Time Environment

For the development, documentation and testing of the above described software artifacts the following software tools and technologies have been used:

- Java SE 1.6.0 as run-time environment for the WCF system
- Eclipse IDE for Java Developers - Indigo SR 1 as development environment and with Subclipse 1.8.4 for SVN versioning and remote backup
- jUNIT 4.8.2v4 for unit testing
- A Debian GNU Linux 6.0 Squeeze 64 bit virtual machine in an Oracle VirtualBox 4.1.12 hypervisor as logically separated R server
- R 2.15.0 64 bit version with R forecast package 3.2 [HK08] and Rserve package 1.7.0 [Urb11] for the forecast processing and statistical evaluations.
- Java REngine library as Java client for Rserve

The computing platform for execution of the above listed software and as environment for the evaluation experiments has been a Sony Vaio z Series VPCZ1 laptop:

- CPU: Intel Core i7 M620 2 Core processor @2.67 GHz featuring hyper-threading and over-boost of frequency up to 3.2 GHz
- OS: Microsoft Windows 7 Ultimate SP1 64bit
- RAM: 8 GB
- HDD: RAID 0 array of 4 solid state disks a 61 GB

4.4.1 Third Party Source Code

For the implementation of the WCF system, source code of the TSLIB Java library has been reused, adapted and extended. The TSLIB Java library is a result of implementation work in the context of a related diploma thesis [Bie12] on the topic of online anomaly detection that has been completed at the Christian-Albrechts University of Kiel by Tillmann Carlos Bielefeld and been supervised by André van Hoorn. The explicit permission has been given to use and adapt the TSLIB code for the concerns of the WCF approach. The TSLIB offers Java classes for the `TimeSeries` data structure, several classes of the `Forecasting` component and the `RServerBridge`. As all of these Java classes needed adaption and extension for the requirements of the WCF system, it has been decided not to simply import and use the TSLIB code. In the `RServerBridge` class unnecessary code has been removed and code added to enable remote Rserve connections. The `TimeSeries` data structure has been debugged and extended to handle frequency, period and capacity attributes. In the `Forecasting` component, the R script of the `AbstractRForecaster`

was extended to provide time series frequency information to the R engine as needed by the complex forecast strategies, and additionally to calculate the MASE estimation to enable an immediate decision which forecast result shows higher accuracy and is written to the WCF system output. The number of available forecast strategies in the `Forecasting` component has been increased by six. All enhancements and removed bugs of the TSLIB have been reported to their primary authors.

4.4.2 RServe Debian Virtual Machine Setup

As it is recommended to execute the server of the RServe package only in an Unix based operation system, a virtual machine has been build that is executable in the Oracle VirtualBox 4.1.12 hypervisor. The virtual machine itself is a Debian GNU Linux 6.0 Squeeze 64 bit configured to run on 2 cores of the described underlying host platform and having assigned 2024 MB of RAM. Only required software packages are installed in this virtual machine to keep its size small (3.5 GB). The R environment for statistical computing is installed in version 2.15.0 for 64 bit operating systems and extended by the R forecast package 3.2 [HK08] and Rserve package 1.7.0 [Urb11]. Rserve has been manually configured to accept remote connections at the default port 6311 on the primary ethernet device without authentication. Via the Unix command `R CMD Rserve` the server is started in deamon-mode and ready to accept remote requests. For safety reasons the Rserve should not be executed by an Unix user with root rights.

Having this configured virtual machine image, brings the advantages of replicability and portability of the WCF system's forecast processing environment. All software installed in this virtual machine is publicly available under the GNU licence.

5. Evaluation

This chapter evaluates the workload classification and forecasting (WCF) approach that is presented in Chapter 3 by conducting experiments using the implementation of the WCF system as outlined in Chapter 4. Different real-world workload intensity behavior traces have been used as input for the conducted experiments to compare particular forecast strategies against the results of the WCF system in various configurations. This chapter starts with a presentation of the used exemplary real-world workload intensity behavior traces in Section 5.1. In Section 5.2, four different experiment scenarios are given together with result illustrations and statistical analysis. The experiments are designed to compare the forecast accuracy of the flexible WCF approach with the accuracy of single forecast strategies in a fair way. In addition, the forecast accuracy is compared to the accuracy that can be achieved just by system monitoring. Finally, an exemplary case study is conduced in Section 5.3 to underline the benefits that are connected to an application of the WCF system. At the end of this chapter in Section 5.4 the experiment results are discussed, as they allow conclusions on how to provide the input data to the WCF system in terms of aggregation level selection or even splitting to achieve optimal and reliable forecast accuracy in particular application scenarios or for certain forecast objectives.

5.1 Exemplary Real-World Workload Traces

As discussed in Section 2.2, real-world workload intensity behavior traces are likely to show strong seasonal patterns in daily periods due to human users of a software service. The daily seasonal patterns are possibly overlaid by patterns of a far longer period like a week or month. Depending on the monitoring precision, a certain noise level is normally observed, but can be reduced by aggregation of monitoring intervals or smoothing techniques as discussed in Section 3.2.7. Deterministic bursts within a trace of a workload intensity behavior are often induced by planned batch tasks on a system that uses the analysed software service (for example in transaction processing systems). In addition, a workload intensity behavior trace can show non-deterministic bursts that cannot be foreseen by any time series analysis technique due to system extrinsic influences as discussed in Section 3.1.

For the evaluation of the WCF approach several different real-world workload intensity behavior traces have been analysed. All of them show the above mentioned characteristics in different peculiarity and collocation. Two of them have been used as input data for the presented experiments:

Wikipedia Germany Page Requests: The hourly number of page requests at the webpage of Wikipedia Germany has been extracted from the publicly available server logs [1] for the October of the year 2011.

CICS Transaction Processing Monitoring data has been provided by IBM for the academic concerns of this thesis. The monitored system has been an IBM z10 mainframe server that offers a CICS transaction processing service. The data covers one week from Monday to Sunday in transaction arrivals per 15 minutes.

5.2 Concept Validation

The following four experiments are designed to compare the WCF forecast result accuracy with the forecast accuracy that can be achieved by constantly applying a fixed forecast strategy chosen from 2.2. In the first experiment the WCF system uses forecast strategies from all four overhead groups and is compared to a fixed use of ETS strategy of the overhead group 3. The following experiments from 2 to 4 then compare the WCF system which is limited to select only strategies from a certain overhead group to the individual strategies of the overhead group in focus. Additionally, in all experiments the naive forecast strategy (which is equal to system monitoring without forecasting) is compared to the other forecast strategies to quantify and illustrate the benefit of applying forecast strategies against just monitoring the arrival rates.

The individual forecasts have been executed with identical forecast objectives and on the same input data. The forecast results are continuous over the simulated period which means that for every observed value there is a forecast mean value and a confidence interval. This experiment design allows an analysis whether the WCF system successfully classifies the workload intensity behavior. A successful classification would mean that the forecast strategy that delivers the highest accuracy for a particular forecast execution is selected by the WCF system in the majority cases using time series analysis and comparisons of suitable subsets of forecast strategies.

To analyse the forecast result accuracy in higher detail, a relative error is calculated for every single forecast point.

$$relativeError_t = \frac{|forecastMeanValue_t - observedArrivalRate_t|}{observedArrivalRate_t}$$

The distributions of these relative errors are illustrated in cumulative histograms which have inclusive error classes on the x-axis and on the y-axis the corresponding percentage of all forecast points. In other words, an $[x; y]$ tuple expresses that y percent of the forecast points have an relative error between 0% and x%. Accordingly, as the constant line $y = 100$ represents the hypothetical optimal distribution with no errors at all, the topmost value of the compared ones is the best for all error classes. The cumulative histogram instead of normal histogram has been chosen to obtain a monotone discrete functions of the error distribution resulting in less intersections with the other error distributions and therefore in a clearer illustration of the data. For better visibility, the histograms are not drawn using pillars but simple lines connecting the single $[x; y]$ tuples.

In addition, statistical key indices like the arithmetic mean, the median and the quartiles as well as the maximum have been computed to enable direct comparison of the relative error distributions and build an order of the forecast strategies according to their achieved forecast accuracy in the experiment scenario.

Finally, directed, paired t-tests from common statistics have been conducted to determine how significantly the average forecast accuracy of a certain strategy is better than the average forecast accuracy of another strategy in comparison of the experiment scenario.

[1] http://dumps.wikimedia.org/other/pagecounts-raw/

5.2.1 Experiment 1: Comparison of WCF, ETS and Naive

The first experiment compares the forecast accuracy of the WCF with the ETS and Naive strategy during a learning process which means that at the beginning there is no historic knowledge available to the individual forecast strategies in comparison. Various strategies of all overhead groups are applied in the WCF forecast strategy during the simulated five days of the CICS workload intensity behavior from Monday until Friday. The forecast horizon (identical for Naive, ETS and WCF) is configured to increase stepwise as given in Table 5.1.

Table 5.1: Experiment 1: Configuration

Experiment Focus	Comparison of unrestricted WCF to static ETS and Naive
Forecast Strategies (overhead group)	WCF(1-4), ETS(3), Naive(1)
Input Data	CICS Monday to Friday, 240 values in transactions per 30 minutes, frequency = 48, 5 periods as days
Horizon (number of forecast points (h) for time series length (tsl))	h = 1 for tsl in [1;24] (1^{st} half period), h = 2 for tsl in [25;144] (until 3^{rd} period complete), h = 12 for tsl in [145;240] (4^{th} and 5^{th} period)

The cumulative error distribution for each of the strategies is shown in Figure 5.2 which demonstrates that the WCF strategy achieves the best forecast accuracy compared to ETS and Naive, as the corresponding line is constantly on the top. The ETS strategy can only partly achieve slightly better forecast accuracy than the Naive strategy though it induces processing overheads of 715 ms per forecast execution compared to 45 ms for the Naive strategy (computation of the confidence intervals). The WCF strategy has an average processing time of 61 ms until the overhead group 4 strategies are selected after the first three periods. For the last two periods the WCF approaches processing time per execution is on average 13.1 seconds. The forecast mean values of the individual strategies and the observation values in the course of time are plotted in Figure 5.1. In this chart it is visible that the ETS strategy forecast mean values have several bursts during the first three periods and therefore do not stay as close to the observed values in a number of cases as the WCF forecast mean values more constantly do. During the last eight forecast executions in the fourth and fifth period (Thursday and Friday) the WCF approach successfully detects the daily pattern of the day before and therefore estimates the course better than the ETS strategy.

Table 5.2: Experiment 1: Result Summary

Strategy	Minimum	25% Quantil	Median	Mean	75% Quantil	Maximum
WCF	0.0128%	9.474%	20.77%	47.39%	49.65%	874.3%
ETS	0.0014%	12.2%	32.31%	75.01%	73.36%	1977%
Naive	0.4917%	16.26%	38.05%	78.88%	81.5%	1671%

In Table 5.2, the error distributions of the individual forecast strategies' accuracies are characterized by basic statistical indices and in addition illustrated in a Box&Whisker plot

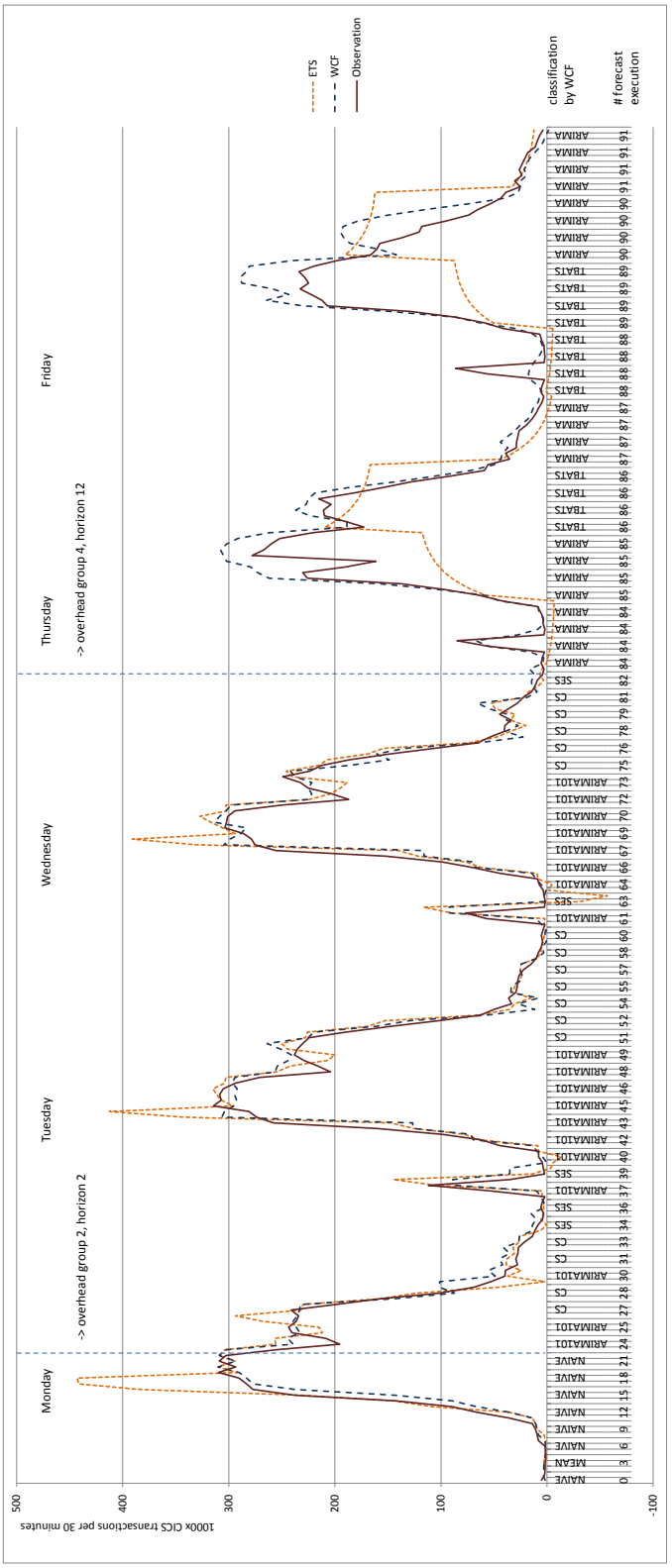
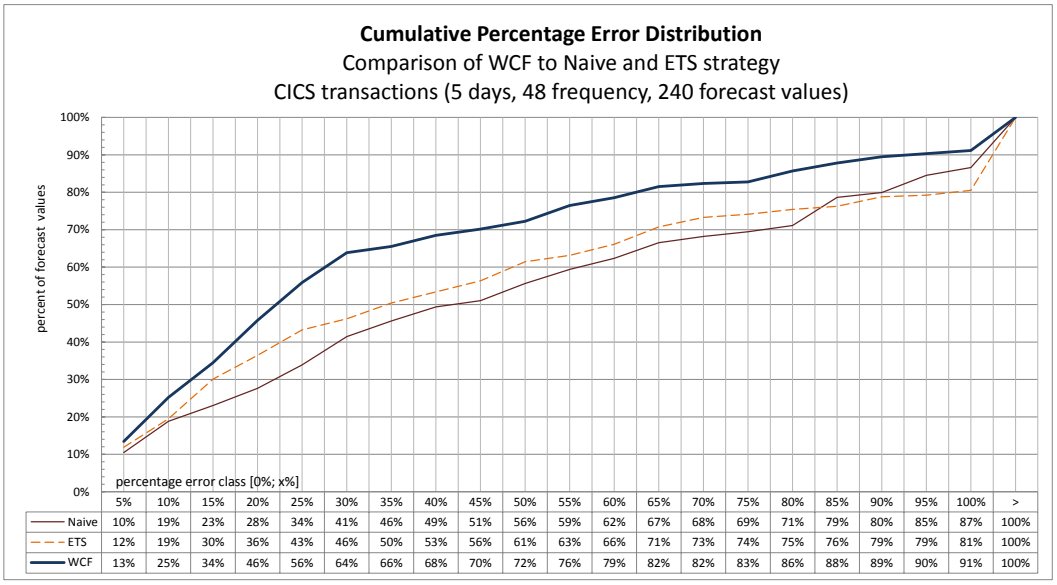


Figure 5.1: Experiment 1: Comparison Chart of WCF and ETS



percentage error class [0%; x%]	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%	>
Naive	10%	19%	23%	28%	34%	41%	46%	49%	51%	56%	59%	62%	67%	68%	69%	71%	79%	80%	85%	87%	100%
ETS	12%	19%	30%	36%	43%	46%	50%	53%	56%	61%	63%	66%	71%	73%	74%	75%	76%	79%	79%	81%	100%
WCF	13%	25%	34%	46%	56%	64%	66%	68%	70%	72%	76%	79%	82%	82%	83%	86%	88%	89%	90%	91%	100%

Figure 5.2: Experiment 1: Cumulative Error Distribution of WCF, ETS and Naive

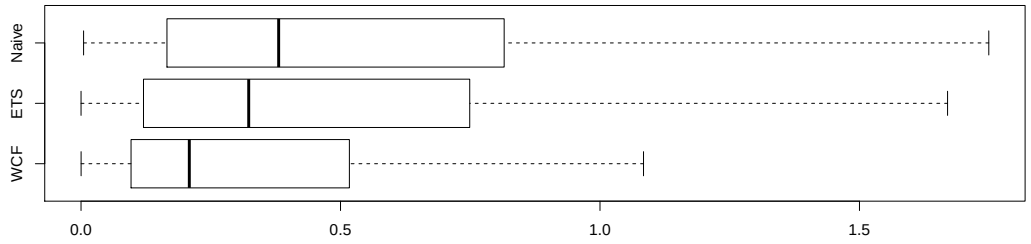


Figure 5.3: Experiment 1: Box&Whisker Plots of the Error Distributions without outliers

in Figure 5.3. The WCF approach shows for example the lowest median value of 20.7% and the lowest mean value of 47.4%. In addition, the WCF apporach has a significantly smaller maximum error value that is important to increase the trust into its forecast mean values.

In Figure 5.4, the WCF and the ETS strategy's percentage error distributions are tested by an paired, directed t-test on the hypothesis that their true difference in means is less than zero. The result says that there is a significant mean of differences of -0.27 indicating that the WCF approach achieves significantly smaller forecast errors. In addition, the WCF and the Naive strategy's percentage error distributions are tested in the same way with the result of a highly significant mean of differences of -0.31 underlining the hypothesis that the application of the WCF approach can be of use compared to just apply system monitoring (which would be equal to the Naive strategy).

Though the ETS strategy is a sophisticated procedure on its own, this experiment shows that the application of this fixed strategy cannot achieve useful forecast mean values as their forecast errors are on average almost as high as of the Naive strategy and show even higher maximal errors.

```
        Paired t-test

data:  WCF and ETS
t = -2.1129, df = 229,
   p-value = 0.01785
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.06031597
sample estimates:
mean of the differences
   -0.2762366
```

```
        Paired t-test

data:  WCF and Naive
t = -2.7639, df = 229,
   p-value = 0.003088
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.1267299
sample estimates:
mean of the differences
   -0.3148823
```

Figure 5.4: Experiment 1: Directed, paired t-test on WCF and ETS error distributions (left) and on WCF and Naive error distributions (right)

5.2.2 Experiment 2: Comparison of Overhead Group 2 Strategies

The second experiment compares the forecast accuracy of the WCF approach with strategies of overhead group 2 forecast and lower (WCF2) against the individual group 2 strategies: CS, ARIMA101 and SES. As a strength of all of these strategies is the trend extrapolation and as none of them is capable to handle seasonal patterns, high forecast accuracy is not likely to be achieved for the CICS workload intensity behavior that shows highly complex daily patterns and therefore only very short term trends are visible. To achieve acceptable forecast accuracy the strategies are applied in a high frequency and with a maximum horizon of only two forecast mean values. Six days of the CICS workload intensity behavior are simulated in this experiment from Monday until Saturday. More details on the experiment configuration are given in Table 5.3.

Table 5.3: Experiment 2: Configuration

Experiment Focus	Comparison of overhead group 2 strategies to WCF restricted to select from group 1 to 2
Forecast Strategies (overhead group)	CubicSpline(2), ARIMA101(2), SES(2), WCF(1-2), Naive(1)
Input Data	CICS Monday to Saturday, 576 values in transactions per 15 minutes, frequency = 96, 6 periods as days
Horizon (number of forecast points (h) for time series length (tsl))	h = 1 for tsl in [1;48] (1^{st} half period), h = 2 for tsl in [49;576] (until 6^{th} period complete),

The cumulative percentage error distribution for each of the executed strategies is given in Figure 5.6. It is visible that the WCF2 approach achieves similar forecast accuracy compared to the SES strategy whereas the ARIMA101 and CS strategies show lower forecast accuracy as they are constantly below the SES and WCF2 lines. As the WCF2 approach is just combining the other strategies internally by the classification and feedback mechanism as in 3.3, it gets visible in this experiment that the WCF2 approaches continuous selection is successful.

All forecast strategies induce approximately the same amount of computational overhead

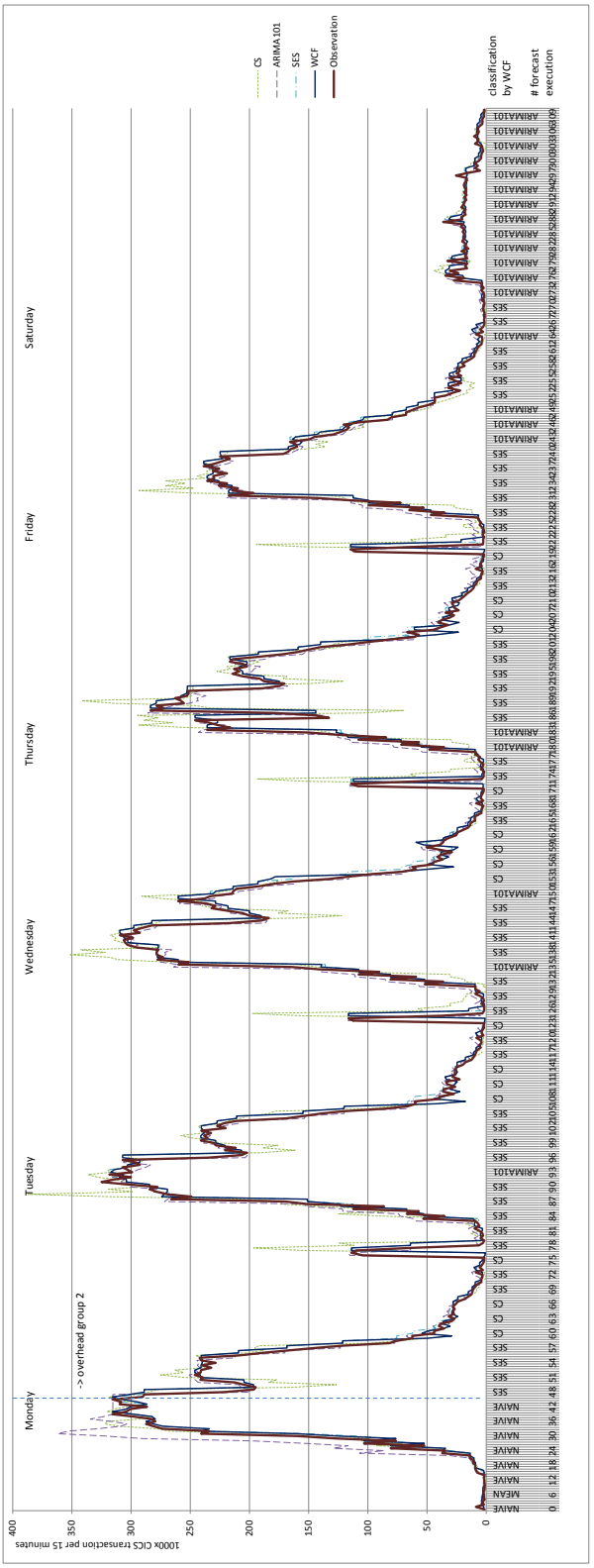
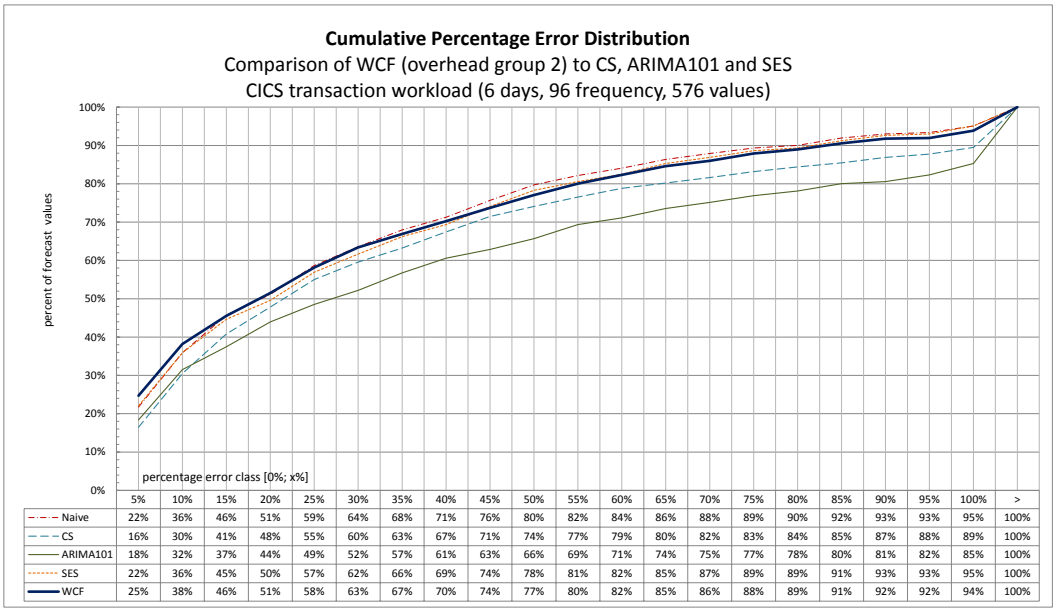


Figure 5.5: Experiment 2: Comparison Chart of WCF2, CS, ARIMA101 and SES



	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%	>
Naive	22%	36%	46%	51%	59%	64%	68%	71%	76%	80%	82%	84%	86%	88%	89%	90%	92%	93%	93%	95%	100%
CS	16%	30%	41%	48%	55%	60%	63%	67%	71%	74%	77%	79%	80%	82%	83%	84%	85%	87%	88%	89%	100%
ARIMA101	18%	32%	37%	44%	49%	52%	57%	61%	63%	66%	69%	71%	74%	75%	77%	78%	80%	81%	82%	85%	100%
SES	22%	36%	45%	50%	57%	62%	66%	69%	74%	78%	81%	82%	85%	87%	89%	89%	91%	93%	93%	95%	100%
WCF	25%	38%	46%	51%	58%	63%	67%	70%	74%	77%	80%	82%	85%	86%	88%	89%	91%	92%	92%	94%	100%

Figure 5.6: Experiment 2: Cumulative Error Distribution of WCF2, CS, ARIMA101 and SES

per execution of only 55 ms on average which allows a high frequency of forecast executions.

The forecast mean values of the individual strategies and the observation values in the course of time are plotted in Figure 5.5. This chart illustrates that the WCF2 strategy constantly stays closer to the observed values than the ARIMA101 strategy (divergences at the beginning) and the CS strategy (which constantly assumes too strong trends at the edges of a seasonal pattern).

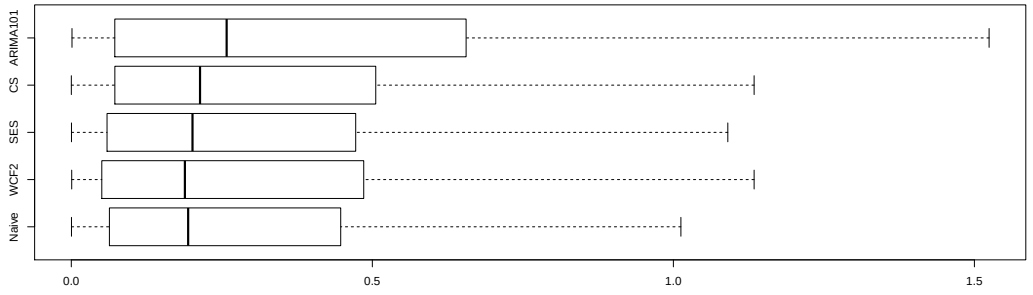


Figure 5.7: Experiment 2: Box&Whisker Plots of the Error Distributions without outliers

In Table 5.4, the error distributions of the individual forecast strategies are characterized by basic statistical indices and in addition illustrated in a Box&Whisker plot in Figure 5.7. On the one hand, the WCF approach shows the lowest median error of 18.6%. But on the other hand, the Naive strategy achieves the lowest mean error of 53.7%. This shows that in presence of strong seasonal patterns the simple trend extrapolating strategies are not useful as their accuracy is comparable or worse than the accuracy of the Naive strategy.

In Figure 5.8, the WCF2 approach and the CS strategy's percentage error distributions are tested by a paired, directed t-test on the hypothesis that their true difference in means is less than zero. The result says that there is a highly significant mean of differences of -0.70 indicating that the WCF approach achieves significantly smaller forecast errors. In addition, the WCF and the ARIMA101 strategy's percentage error distributions are tested in the same way with the result of a highly significant mean of differences of -0.17. These

Table 5.4: Experiment 2: Result Summary

Strategy	Minimum	25% Quantil	Median	Mean	75% Quantil	Maximum
WCF2	0.067%	5.063%	18.64%	59.69%	47.94%	3777%
Naive	0.029%	6.215%	19.26%	53.79%	44%	3779%
SES	0.029%	5.935%	20.13%	54.8%	47.24%	3777%
CS	0.005%	7.232%	21.38%	130.4%	50.57%	6476%
ARIMA101	0.114%	7.223%	25.68%	77.39%	65.5%	3776%

```
        Paired t-test

data:  WCF2 and CS
t = -4.8649, df = 570,
   p-value = 7.418e-07
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.4675543
sample estimates:
mean of the differences
   -0.7069751
```

```
        Paired t-test

data:  WCF2 and ARIMA101
t = -5.7216, df = 570,
   p-value = 8.529e-09
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.1260808
sample estimates:
mean of the differences
   -0.177067
```

Figure 5.8: Experiment 2: Directed, paired t-test on WCF2 and CS error distributions (left) and on WCF2 and ARIMA101 error distributions (right)

two results underline that the WCF2 approach achieves higher forecast accuracy by its classification mechanism than the two other strategies (CS and ARIMA101) on their own.

As the SES strategy achieved the highest accuracy in this experiment, a high similarity of the WCF2 approaches and the SES strategy's percentage error distributions is expected and accordingly a significant difference of means cannot be detected by a paired t-test as in Figure 5.9. When comparing the WCF2 approaches and the Naive strategies' error distributions, a small but significant mean of differences of 0.058 is detected by the paired, directed t-test. In this case the Naive strategy is as good as the WCF2 approach. This may change if the workload intensity behavior does not show that strong seasonal patterns for example in data of higher resolution (seconds, minutes) for short term trend interpolation or on highly aggregated data for long term trend extrapolation (weeks, months, years). This has not yet been validated as no high resolution or long term real-world workload intensity behavior data is available up to now.

```
        Paired t-test

data:  WCF2 and SES
t = 1.9554, df = 570,
   p-value = 0.9745
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf 0.09009366
sample estimates:
mean of the differences
   0.04889562
```

```
        Paired t-test

data:  WCF2 and Naive
t = 2.3452, df = 570,
   p-value = 0.00968
alternative hypothesis:
   true difference in means
   is greater than 0
95 percent confidence interval:
   0.0175491       Inf
sample estimates:
mean of the differences
   0.05899171
```

Figure 5.9: Experiment 2: Directed, paired t-test on WCF2 and SES error distributions (left) and on WCF2 and Naive error distributions (right)

5.2.3 Experiment 3: Comparison of Overhead Group 3 Strategies

The third experiment compares the forecast accuracy of the WCF approach using overhead group 3 forecast strategies (WCF3) against the individual group 3 strategies: ETS and tBATS. As a strength of both of these strategies is the seasonal pattern detection, a historic knowledge of 3 periods is given at the beginning. These strategies do not need to be applied in a high frequency as they can deliver high forecast accuracy for a longer horizon. 18 days of the Wikipedia workload intensity behavior are simulated in this experiment from Thursday until the next but one Sunday. More details on the experiment configuration are given in Table 5.5.

Table 5.5: Experiment 3: Configuration

Experiment Focus	Comparison of overhead group 3 strategies to WCF restricted to select from group 3
Forecast Strategies (overhead group)	ETS(3), tBATS(3), WCF(3), Naive(1)
Input Data	Wikipedia 3 weeks, 504 values in page requests per hour, frequency = 24, 21 periods as days, first 3 days Monday to Wednesday as historic knowledge
Horizon (number of forecast points (h) for time series length (tsl))	h = 12 for tsl in [73;504] (4^{th} until 21^{st} period), no forecasts for the first 3 periods

The cumulative percentage error distribution for each of the executed strategies is given in Figure 5.11. The tBATS strategy and the WCF3 approach achieve both a similar forecast accuracy. The ETS strategy is only slightly worse. The big gap to the Naive strategy indicates that the forecast mean values of these complex forecast strategies are possibly useful for resource planning in this context.

All three forecast strategies induce approximately the same amount of computational overhead as they belong to the same overhead group: tBATS on average 10 seconds per execution, ETS 13.8 seconds and WCF3 19 seconds (as it executes both tBATS and ETS

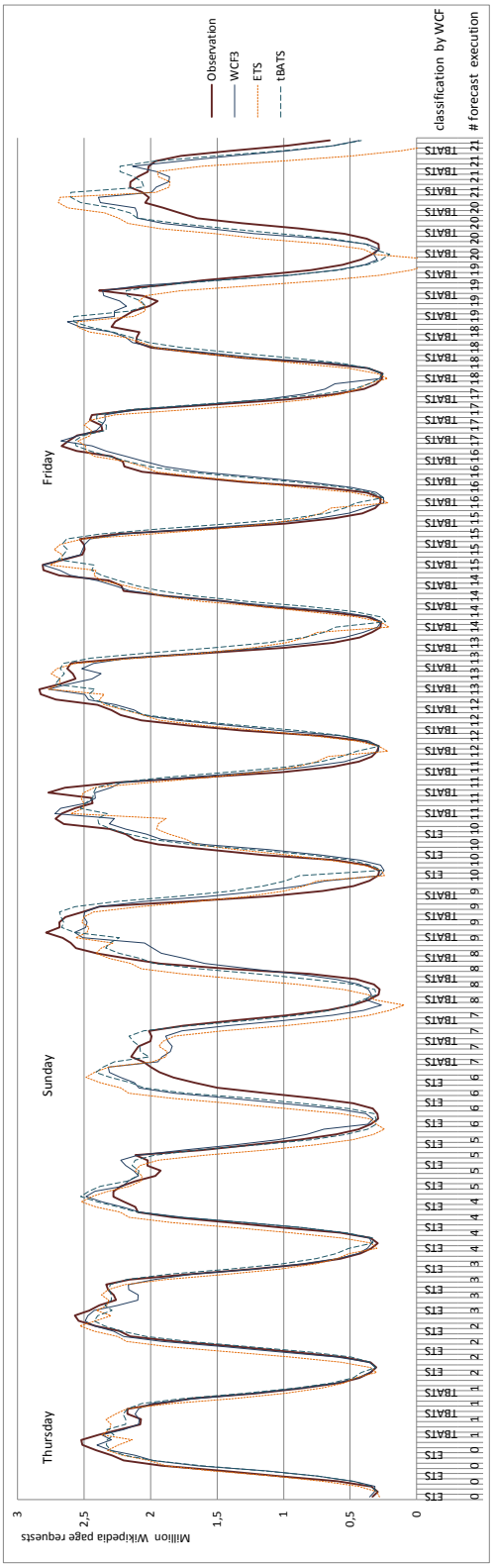


Figure 5.10: Experiment 3: Comparison Chart of WCF3, ETS, tBATS and ETS

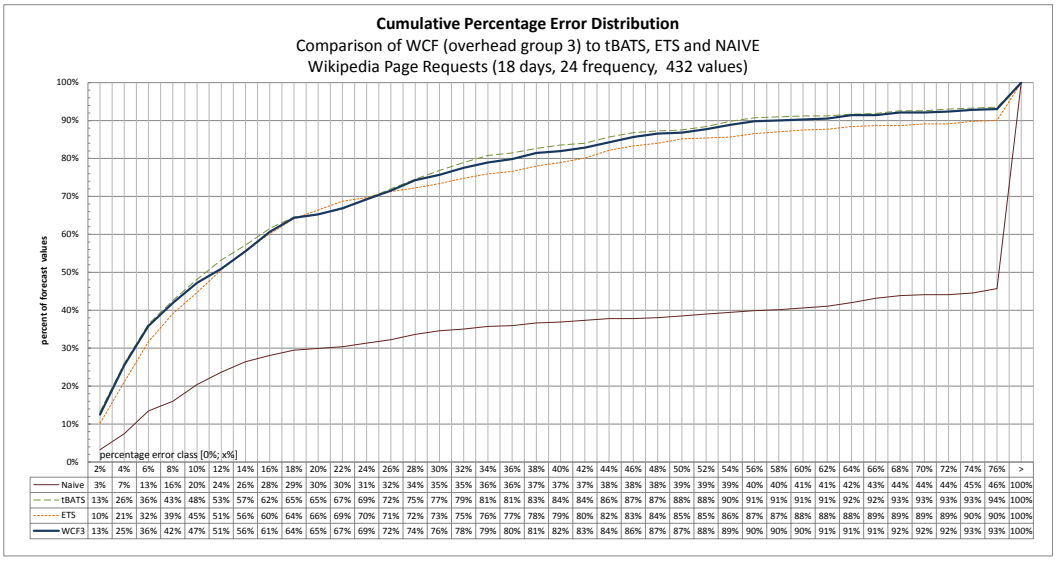


Figure 5.11: Experiment 3: Cumulative Error Distribution of WCF3, ETS, tBATS and Naive

every second time). Their computations are based on the last seven observed periods and therefore on a maximum number of 168 observation values.

The forecast mean values of the individual strategies and the observation values in the course of time are plotted in Figure 5.10 only for the first 10 days of the simulated 18 days. This chart illustrates the high forecast accuracy of all three forecast strategies especially at the edges of the daily seasonal patterns. It can be seen that the changes in the amplitudes of the daily patterns for example from Sunday to Monday induce higher forecast errors than for constant amplitudes on working days.

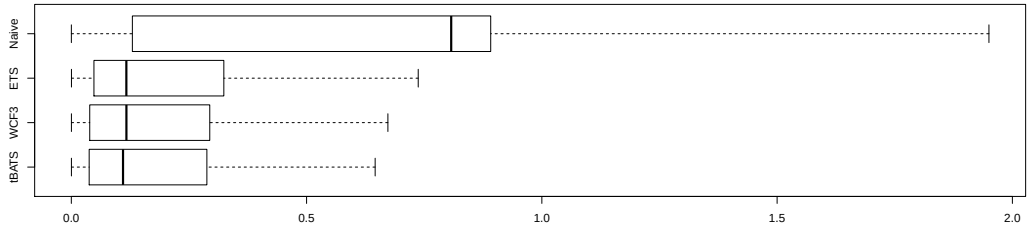


Figure 5.12: Experiment 3: Box&Whisker Plots of the Error Distributions without outliers

Table 5.6: Experiment 3: Result Summary

Strategy	Minimum	25% Quantil	Median	Mean	75% Quantil	Maximum
tBATS	0.0209%	3.793%	10.93%	22.76%	28.33%	185.2%
WCF3	0.0209%	3.915%	11.63%	24.01%	29.42%	209.6%
ETS	0.0294%	4.807%	11.62%	27.44%	32.41%	241.2%
Naive	0.0081%	12.98%	80.73%	127.4%	89.13%	1013%

In Table 5.6, the error distributions of the individual forecast strategies are characterized by basic statistical indices and in addition illustrated in a Box&Whisker plot in Figure 5.12. The tBATS strategy shows the lowest median value of only 10.9%. The low maximum

errors of all three forecast strategies strengthen the trustworthiness of the forecast mean values compared to the naive strategy. As the WCF3 approach is a combination of the two other strategies, it obviously cannot be better than these for individual executions but their error values are more close to the in this case better performing tBATS strategy.

```
        Paired t-test

data:  WCF3 and ETS
t = -4.1869, df = 430,
   p-value = 1.716e-05
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.02077467
sample estimates:
mean of the differences
   -0.03426498
```

```
        Paired t-test

data:  WCF3 and tBATS
t = 1.1914, df = 430,
   p-value = 0.2341
alternative hypothesis:
   true difference in means
   is not equal to 0
95 percent confidence interval:
   -0.008086287  0.032979603
sample estimates:
mean of the differences
   0.01244666
```

Figure 5.13: Experiment 3: Directed, paired t-test on WCF3 and ETS error distributions (left) and on WCF3 and tBATS error distributions (right)

When comparing the WCF3 approaches and the ETS strategy's error distributions by a paired, directed t-test as in Figure 5.13, the detected mean of differences is small with a value of -0.034 but highly significant which means that the WCF3 approach constantly achieves a slightly higher accuracy. The mean of differences between the WCF3 approaches and tBATS strategy's error distributions is even smaller and not significant as the WCF3 selects in this scenario the tBATS strategy in about 80% of the forecast executions . This shows that the WCF3 approach is able to continuously choose the better performing strategy.

5.2.4 Experiment 4: Comparison of Overhead Group 4 Strategies

The fourth and last presented experiment compares the forecast accuracy of the WCF approach using overhead group 4 forecast strategies (WCF4) against the individual group 4 strategies: ARIMA and tBATS. This is the only difference to the configuration in experiment 3. The configuration of experiment 4 is summarized in the Table 5.7.

Table 5.7: Experiment 4: Configuration

Experiment Focus	Comparison of overhead group 4 strategies to WCF restricted to select from group 4
Forecast Strategies (overhead group)	ARIMA(4), tBATS(4), WCF(4), Naive(1)
Input Data	Wikipedia 3 weeks, 504 values in page requests per hour, frequency = 24, 21 periods as days, first 3 days Monday to Wednesday as historic knowledge
Horizon (number of forecast points (h) for time series length (tsl))	h = 12 for tsl in [73;504] (4^{th} until 21^{st} period), no forecasts for the first 3 periods

The cumulative percentage error distribution for each of the executed strategies is given in Figure 5.15. The ARIMA strategy and the WCF4 approach achieve both a similar forecast accuracy. The tBATS strategy is only slightly worse. Again, as in experiment 3, the big gap to the Naive strategy indicates that the forecast mean values of these complex forecast strategies are of possible use for resource planning in this context. In addition, this chart illustrates the even higher forecast accuracy of the WCF4 approach and ARIMA strategy compared to WCF3 approach in experiment 3.

The computational overhead per forecast execution are on average 22 seconds for the WCF4 approach, again 10 seconds for the tBATS strategy. The ARIMA strategy needed on average 15 seconds per forecast execution but the durations show a higher variance and maximal duration of 56 seconds. As in experiment 3, their computations are based on the last seven observed periods and therefore on a maximum number of 168 observation values. These measurements underline the correctness of the presented overhead grouping of the forecast strategies in Table 3.2.

The forecast mean values of the individual strategies and the observation values in the course of time are plotted in Figure 5.14 only for the first 10 days of the simulated 18 days. Their shapes show similar characteristics as in experiment 3 but even closer estimations of the pattern amplitudes.

In Table 5.8, the error distributions of the individual forecast strategies are characterized by basic statistical indices and in addition illustrated in a Box&Whisker plot in Figure 5.16. The ARIMA strategy shows the lowest median value of only 9.6%. The WCF4 approach has the lowest mean error values and achieves this improvement by internally executing the better performing strategy. In this scenario, both strategies are selected with a similar probability resulting in a combination of their strength for particular situations. In addition, the WCF4 approach has the lowest maximum error and therefore the highest trustworthiness of the compared strategies. These experiment results show again that an measurable improvement can be achieved by the WCF approach.

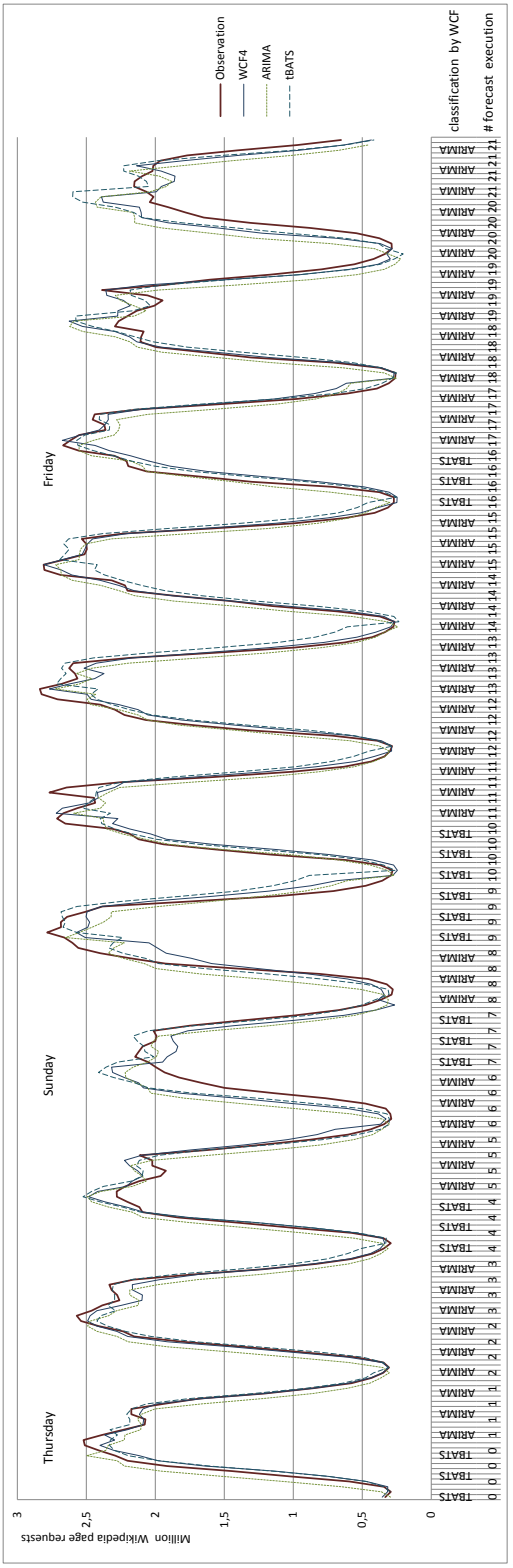


Figure 5.14: Experiment 4: Comparison Chart of WCF4, tBATS, ARIMA

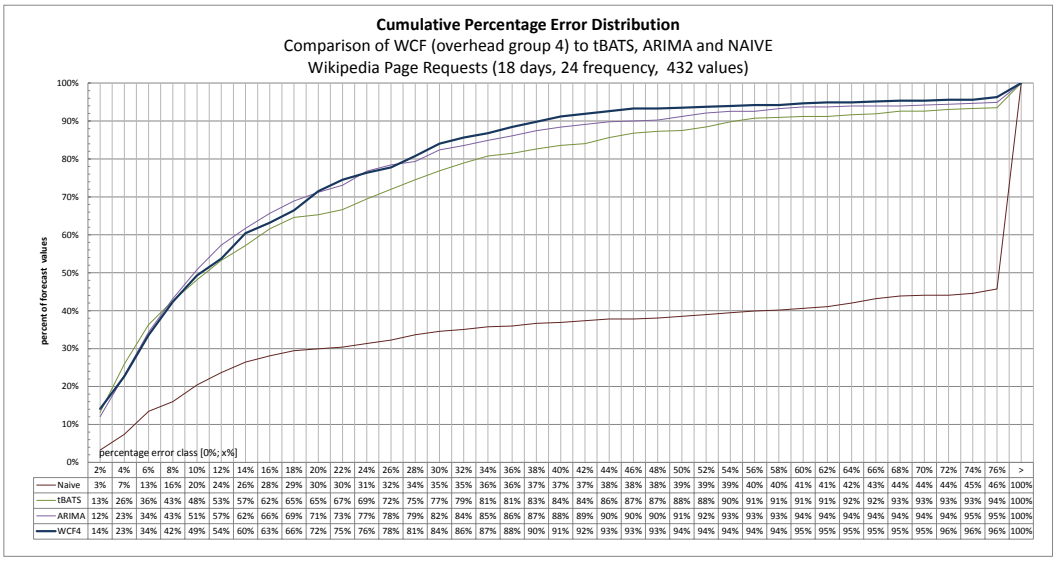


Figure 5.15: Experiment 4: Cumulative Error Distribution of WCF4, tBATS, ARIMA and Naive

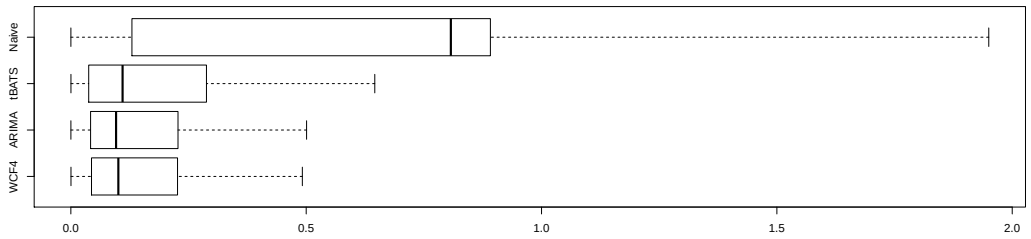


Figure 5.16: Experiment 4: Box&Whisker Plots of the Error Distributions without outliers

In Figure 5.17, a comparison of the WCF4 approach with both others strategies, the ARIMA and the tBATS, by the paired, directed t-tests detects a highly significant mean of differences within percentage error distributions in both cases. This indicates that the WCF4 approach is able to correctly select the strategy that is more likely to show higher accuracy in particular cases out of the two others strategies ARIMA and tBATS. Therefore the WCF4 approach is able to combine their strengths.

If the directed, paired t-test is used to compare the error distributions of the WCF4 and the WCF3 (experiment 3) as in Figure 5.18, a highly significant mean of differences of -0.65 is found. The WCF4 approaches internal use of the overhead group 4 strategy ARIMA has obviously caused this improvement in accuracy of WCF4 compared to WCF3. The mean of differences of -1.1 is detected when comparing the WCF4 approach to the

Table 5.8: Experiment 4: Result Summary

Strategy	Minimum	25% Quantil	Median	Mean	75% Quantil	Maximum
WCF4	0.0272%	4.389%	10.12%	17.49%	22.67%	125.5%
ARIMA	0.0096%	4.193%	9.608%	19.89%	22.77%	340.6%
tBATS	0.0209%	3.793%	10.93%	22.76%	28.33%	185.2%
Naive	0.0081%	12.98%	80.73%	127.4%	89.13%	1013%

```
        Paired t-test

data:  WCF4 and tBATS
t = -4.2109, df = 430,
   p-value = 1.55e-05
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.03208695
sample estimates:
mean of the differences
   -0.05272829
```

```
        Paired t-test

data:  WCF4 and ARIMA
t = -1.843, df = 430,
   p-value = 0.03301
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.002530368
sample estimates:
mean of the differences
   -0.02396612
```

Figure 5.17: Experiment 4: Directed, paired t-test on WCF4 and tBATS error distributions (left) and on WCF4 and ARIMA error distributions (right)

```
   Paired t-test

data:  WCF4 and WCF3
t = -5.255, df = 430,
   p-value = 1.168e-07
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.04473058
sample estimates:
mean of the differences
   -0.06517494
```

```
        Paired t-test

data:  WCF4 and Naive
t = -11.6577, df = 430,
   p-value < 2.2e-16
alternative hypothesis:
   true difference in means
   is less than 0
95 percent confidence interval:
   -Inf -0.9436938
sample estimates:
mean of the differences
   -1.099108
```

Figure 5.18: Experiment 4: Directed, paired t-test on WCF4 and WCF3 error distributions (left) and on WCF4 and Naive error distributions (right)

Naive strategy by the directed, paired t-test. This enormous mean of differences of the percentage errors indicates the high potential of the WCF4 approach in this scenario to deliver forecast results that can be of use for a resource planning system, which is further analysed in the presented case study in the following section.

5.3 Case Study

The following case study illustrates how the WCF approach can be used to plan resources of a software service. The WCF offers a higher degree of flexibility due to the spectrum of integrated forecast strategies. None of these forecast strategies can offer these degrees of flexibility on their own. On the one hand, the overhead group 2 strategies do not achieve a good accuracy even if three periods of data are available and on the other hand, the overhead group 3 and 4 strategies cannot return a forecast result if only a few values are available or be applied in a high frequency due to their computational overheads.

In the presented scenario, there is no historic knowledge available at the beginning and the WCF has to adapt to this and wait for the first 3 periods until the overhead group 4 strategies can be applied. It is assumed that the underlying system whose workload intensity behavior is analysed is linearly scalable and has two known thresholds at different levels. For an arrival rate higher than a threshold values, the SLA of the average response time is violated. It is assumed that the system's resource planner reacts on these SLA violations and adds additional computing resources, for example starts a new server instance. For an arrival rate lower than a threshold value, the running server instances are not efficiently used and therefore one of them is stopped.

In this scenario, the WCF system provides forecast mean values and confidence intervals to the resource planer of the system which then can proactively add or remove server instances at that point in time, when the resource or server is needed, in addition to solely reacting on SLA violations.

Details on the WCF configuration are given in Table 5.9.

Table 5.9: Case Study: WCF System Configuration and Input Data

Forecast Strategy (overhead group)	WCF(1-4)
Input Data	Wikipedia 3 weeks, 504 values in page requests per hour, frequency = 24, 21 periods as days
Horizon (number of forecast points (h) for time series length (tsl))	h = 1 for tsl in [1;12] (1^{st} half period) h = 3 for tsl in [13;72] (until 3^{rd} period complete) h = 12 for tsl in [73;504] (4^{th} until 21^{st} period),

In the chart in Figure 5.19, the WCF forecast values are plotted together with the corresponding confidence intervals and the observed values. The two dotted lines represent the thresholds that define when a server instance needs to be started or stopped. The upper threshold is placed in a way that it is not reached constantly in every daily seasonal period (for example not on the weekends).

The starts and stops of server instances are triggered either reactive after an SLA violation or in a planned way anticipating the points in time, when an SLA violation would happen. Table 5.10 depicts the single points, when the thresholds are crossed. It can be seen that the SLA violations cannot be forecast in the first three daily periods besides 2 cases with a small decision window as defined in Section 2.4. For the following daily periods the SLA violations can be correctly anticipated in the majority of cases. Only the when the amplitude of the daily pattern changes for example before and after the weekends the forecast mean values deliver false positives or do not anticipate correctly the need for an additional computing resource.

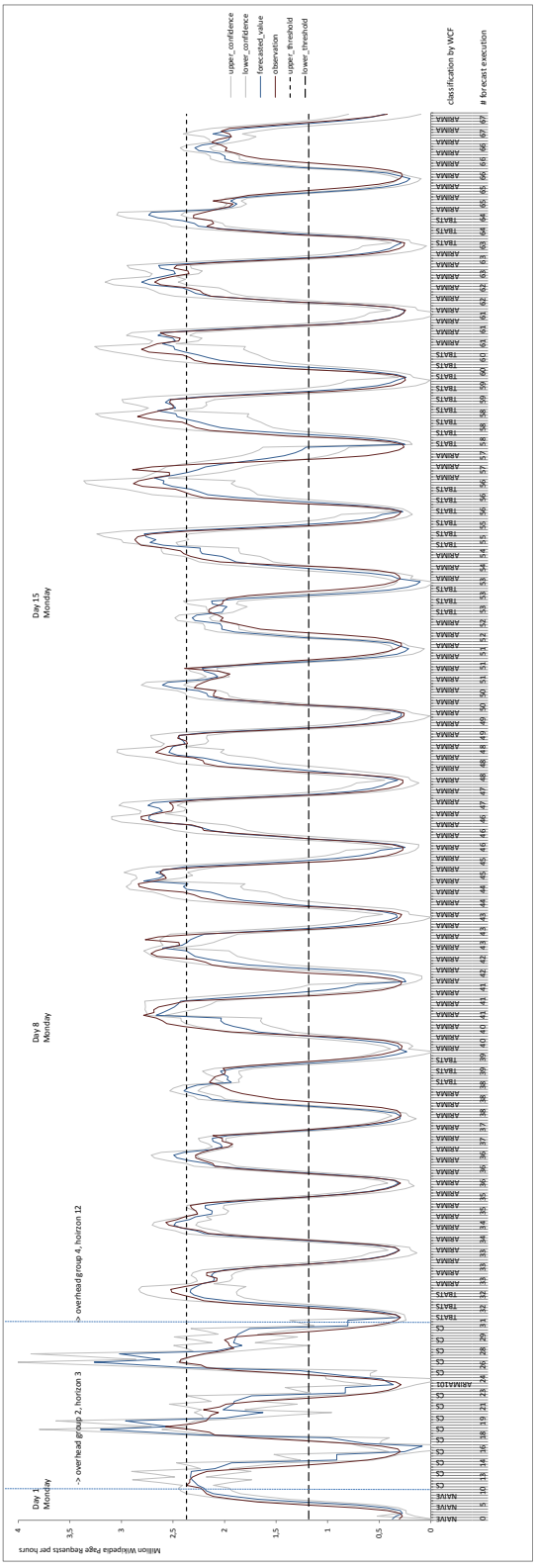


Figure 5.19: Case Study Chart: 21 days Wikipedia Page Requests, WCF4 approach

Table 5.10: Case Study: Table of Prevented SLA Violations

	upper_threshold: 2 or 3 server instances				lower_threshhold: 1 or 2 server instances		
day	cut	direction	action type	critics	direction	action type	critics
1	no		none		up	reactive	not detected
1	no		none		down	reactive	not detected
2	yes	up	proactive	short provisioning time	up	reactive	not detected
2	yes	down	reactive	not detected	down	reactive	not detected
3	yes	up	proactive	short provisioning time	up	reactive	not detected
3	yes	down	reactive	not detected	down	reactive	not detected
4	yes	up	reactive	not detected	up	proactive	
4	yes	down	reactive	not detected	down	proactive	
5	yes	up	proactive		up	proactive	
5	yes	down	proactive		down	proactive	
6	no	up	proactive	false positive	up	proactive	
6	no	down	proactive	false positive	down	proactive	
7	no		none		up	proactive	slightly too early
7	no		none		down	proactive	slightly too early
8	yes	up	reactive	not detected	up	proactive	
8	yes	down	proactive		down	proactive	slightly too late
9	yes	up	proactive	slightly too late	up	proactive	slightly too late
9	yes	down	proactive	slightly too early	down	proactive	
10	yes	up	proactive	slightly too late	up	proactive	
10	yes	down	proactive		down	proactive	
11	yes	up	proactive		up	proactive	
11	yes	down	proactive		down	proactive	
12	yes	up	proactive		up	proactive	
12	yes	down	proactive		down	proactive	
13	no	up	proactive	false positive	up	proactive	
13	no	down	proactive	false positive	down	proactive	
14	no		none		up	proactive	slightly too early
14	no		none		down	proactive	slightly too early
15	yes	up	reactive	not detected	up	proactive	
15	yes	down	proactive		down	proactive	
16	yes	up	proactive	slightly too late	up	proactive	
16	yes	down	proactive	slightly too early	down	reactive	not detected
17	yes	up	proactive	slightly too late	up	proactive	
17	yes	down	proactive		down	proactive	
18	yes	up	proactive		up	proactive	
18	yes	down	proactive		down	proactive	
19	yes	up	proactive		up	proactive	
19	yes	down	proactive		down	proactive	
20	no	up	proactive	false positive	up	proactive	
20	no	down	proactive	false positive	down	proactive	
21	no		none		up	proactive	slightly too early
21	no		none		down	proactive	

Table 5.11 summarizes all detected SLA violations: In the worst case 36 and in the best case 23 SLA violations would have been monitored when using the forecast mean values by the WCF system instead of 78 when just reacting on SLA violations. 16.7% of the SLA violations could not be anticipated by the WCF system in addition to 7.7% false positives in this scenario.

In Table 5.8, the error distribution of the WCF forecast approach is characterized by basic statistical indices. Half of the forecast mean values have an relative error below 9.4% and even-quarter of them have an relative error below 22.9%. This low average of relative errors with a small variance builds up a high level of trust into the forecast mean values. The reason for the high maximum relative errors are the learning periods at the beginning.

Table 5.11: Case Study: SLA Violations Summary

Comparison	Reactive vs. Proactive using WCF
Purely reactive	78 SLA violations
WCF proactive	23 best case 36 worst case SLA violations
WCF false positives	6 of 78 = 7.7%
WCF not detected	13 of 78 = 16.7%

Table 5.12: Case Study: Forecast Error Distribution

	Minimum	25% Quantil	Median	Mean	75% Quantil	Maximum
WCF4	0.0041%	4.145%	9.4%	20.47%	22.92%	365.3%

5.4 Experiment Result Interpretation

The presented experiments demonstrate that the WCF system is able to sensitively select a suited forecast strategy for particular situations thereby improving the overall forecast accuracy and reducing the number of outliers as compared to a single forecast strategy applied repeatedly in a static way. As demonstrated in the case study, the interpretation of WCF forecast results by proactive resource provisioning reduces the number of SLA violations by between 52% to 70%. In addition, the automated, flexible and dynamic forecast strategy selection of the WCF's workload intensity behavior classification approach supports the system user to select a forecast strategy according to given objectives and not yet monitored data. Especially at the beginning of a workload intensity behavior's life time, when no or few historic data is available, a static decision would not fit for the workload's lifetime. With its dynamic design and the flexibility to react on changes in the workload's intensity behavior, the WCF system is able to adapt to these changes, thereby increasing the accuracy and reliability of the forecast results.

The WCF system enables online and continuous forecast processing with controllable computational overheads. To achieve this, forecast strategy executions and workload intensity behavior classifications are scheduled in configurable periods by the WCF system. In all experiments, the processing times of all forecast strategy stayed within the boundaries given by their corresponding overhead group. This shows on the one hand that the grouping of strategies given in Section 3.2 is suitable. On the other hand, this enables the WCF system to schedule the individual executions assuring that the forecast results are available before their corresponding arrival rates are monitored. This is crucial especially for the executions of forecasts in a high frequency with short horizons as for example in Experiment 2. The WCF system warns its user if the result has not been provided in time to indicate that further processing of this result would not be useful. During all presented experiments and the case study this warning has not been observed, though the simulated time within the experiments (up to three weeks) has been mapped to an experiment execution duration of less than 60 minutes. This indicates that the WCF system is suitable for online usage as it provides continuous forecast results under controllable computational overheads.

Concerning the results of experiment 2, the trend extrapolating strategies in overhead group 2 cannot perform as good as the overhead group 3 and 4 strategies even when

executed in high frequency and therefore do not generate a value for resource planning in these scenarios. This is due to the fact that all analysed data on real-world workload intensity behaviors show strong daily seasonal patterns. Due to a lack of data with no strong seasonal components, it is left for future work to analyse whether these strategies perform better on high resolution data (seconds, minutes) or for highly aggregated long term data (months, years) when stronger trends are visible and overhead group 3 and 4 strategies would induce unaffordable computing overheads.

Furthermore, the experiments 1, 3 and 4 showed that if the overhead group 3 and 4 strategies get at least 3 periods of data, they achieve a high forecast accuracy. But providing more periods does not improve forecast accuracy but rises the computational overheads of these strategies significantly. The shape of three periods can be easily described by less than 200 arrival rate values without loosing too much precision by aggregating arrival rate values. This is the number of time series values held in memory for computations connected to classification and forecasts. By sticking to this limitation in time series length, the overhead group 3 and 4 strategies stay within their estimated computation times.

For a workload intensity behavior with strong daily seasonal pattern and high amplitude variance due to known calendar effects, the forecast accuracy might be strongly improved by splitting this workload intensity behavior into two separated time series: regular working days in the first and in the second weekends together with public holidays. This can reduce the impact of the possibly strong overlay of weekly patterns. For future work, it is planned to automatically support such time series splittings and the selection of the data aggregation level for varying forecast objectives. This can be achieved by a realisation of an intelligent filter applied to the monitoring data before is provided in form of time series to the WCF system.

6. Related Work

In the foundations in Section 2, four groups of related work are mentioned that do not include directly comparable work.

The first group focuses the field of workload classification in general without focus on forecasting techniques [CS85, AW96, SWHB06, vHRH08] as presented more detailed at the beginning of Section 2.2.

The second group includes all considered publications on forecasting approaches of the time series analysis that have been incorporated in the WCF approach of this thesis. The standard works of the time series analysis [Hyn08, BJR08, Shu11] and research papers on individual approaches [GKO+08, SH05, HK08, HKPB02, DLHS11, Goo10] are introduced in Section 2.3.2.

In Section 2.3.1, the third group of related work is summarized, as these publications focus in techniques for pattern identification in time series data [Ols01, WL05, ZLDL11, Raa93, SGL+11, ALSS95, HB10].

The fourth group of related work focuses on architectures for reactive resource provisioning mechanisms as in [MHL+11, JHJ+10, HBK11, CEM+10, ACCM09] as well as on the evaluation of the potentials for cost and energy savings by elasticity of resources in [OAL11, AMM+09, TBL09, OLG10, LO10] as mentioned in the introduction in Section 1.

Comparable related work is divided into three groups:

The first group of related work considers research publications that focus on an evaluation of forecast methods applied to workload-related or performance monitoring data. In 2004, Bennani and Menasce have published their research results of an robustness assessment on self managing computer systems under highly variable workloads in [BM04] where they come to the conclusion that proactive resource management improves a system's robustness under highly variable workloads. In addition, the authors contribute by a comparison of three different trend interpolating forecast methods (polynomial interpolation, weighted moving averages and exponential smoothing) that have been introduced as means for workload forecasting in Menasces and Almeida's book [MA98] on "Capacity Planning for Web Performance - Metrics, Models and Methods" published in 1998. Menasce and Bennani propose to select the forecast methods according to the lowest R^2 error as feedback. Even though the focus of this related work lays more on the potentials of proactive resource provisioning and the evaluation of forecast methods is limited to basic trend interpolation strategies without any pattern recognition for seasonal time series components, the

authors' idea to use the feedback of an accuracy metric for forecast strategy selection is central the approach in this thesis. In another case of related work, Frotscher assesses in his bachelor thesis [Fro11] the capabilities to predict response times of two concrete and simple ARIMA models without seasonality, simple exponential smoothing and the Holt-Winters approach as a special case of the extended exponential smoothing. The evaluation is based on generated and therefore possibly unrealistic times series data. The author admits that the spectrum of forecast methods offered by time series analysis is not covered and is critical about the capability to predict response times of the evaluated methods as their strengths in trend extrapolation does not suit to typically quickly alternating response times.

A second group of related work considers approaches for workload forecasting that do not base on the methods of time series analysis as alternatives to WCF approach in this thesis. For example in Kleeberg's diploma thesis [Kle] and a related research paper [BBR+07], it is proposed to use neuronal nets and machine learning approaches for demand prediction. This demand predictions are meant to be used by an operating system's resource manager. Goldszmidt, Cohen and Powers use an approach based on Bayesian learning mechanisms for feature selection and short term performance forecasts described in [GCP05]. In these cases of related research, the training of the applied neuronal nets on a certain pattern need to be completed before the nets can provide pattern based forecasts. This stepwise procedure limits the flexibility and implies the availability of huge amounts of monitoring data for the mining of possibly observable patterns.

The third group of related work covers research that has its focus on approaches for proactive resource provisioning and make use of tailored forecast methods. The authors of the publication [GRCK07] use a tailored method to decompose a time series by into its dominating frequencies using a Fourier transformation and trend interpolation techniques to generate a synthetic workload as forecast. Similarly in [HMBN10], Fourier analysis techniques are applied to predict future arrival rates. In [CDM11] the authors base their forecast technique on pattern matching methods to detect non-periodic repetitive behavior of cloud clients. The research of Bobroff, Kochut and Beaty presented in [BKB07] has its focus on the dynamic placement of virtual machines, but workload forecasting is covered by the application of static ARMA processes for demand prediction. In [KKK10] an approach is proposed and evaluated that classifies the gradients of a sliding window as a trend estimation, on which the resource provisioning decision are then based. The authors Grunske, Aymin and Colman of [AGC12, ACG12] focus on QoS forecasting such as response time and propose an automated and scenario specific enhanced forecast approach that uses a combination of ARMA and GARCH stochastic process modeling framework for frequency based representation of time series data. This way, the authors achieve improved forecast accuracy for QoS attributes that typically have quickly alternating values.

A common limitation of related research work in this group is the focus on single strategies that are optimised or designed to cover a subset of typical situations and are therefore not able to cover all possible situations adequately as it can be achieved by an combination of the innovation state space frameworks of ETS and tBATS and the auto-regressive integrated moving averages (ARIMA) framework for stochastic process modeling.

In the research work that is presented in [MIK+10], different methods of the time series analysis and Bayesian learning approaches are applied and periodically selected by their forecast accuracy. It is not evaluated how significantly this feedback improves the overall forecast accuracy and there is no information given on how the user's forecast objectives are captured and on how the computational overheads can be controlled. In contrast to this thesis, the authors concentrate on the prediction of resource consumptions.

In most cases of related research work besides [HMBN10], the resource utilisation or average response times are monitored and taken as an indirect metric for the recent workload intensity, but these performance metrics are indirectly influence by the changing amounts of provisioned resources and other factors. The forecasts are then based on these values that bypass the resource demand estimation per request and interleave by principle performance relevant characteristics of the software system as they can be modelled in a system performance model.

7. Conclusion

Today's resource managing systems of virtualized computing environments often work solely reactive by using thresholds based rules, whereas the potential of proactive resource provisioning to improve a system's performance and efficiency are not leveraged yet. The results of this diploma thesis can be seen as a step towards making use of this potential by being able to compute continuous and reliable forecasts of workload intensity behaviors with appropriate accuracy.

In this thesis, the first step to achieve this has been the identification of workload intensity behavior specific characteristics and metrics that quantify them. Secondly, it has been a crucial point to establish a survey on the strengths, weaknesses and requirements of existing forecast strategies from the time series analysis and ways to estimate and evaluate the forecast accuracy of individual forecast executions. Thirdly, an approach has been introduced that is able to classify a workload intensity behavior and to dynamically select an appropriate forecast strategy. This has been achieved by using direct feedback mechanisms that evaluate and compare the recent accuracy of different forecast strategies incorporated into a decision tree that considers given forecast objectives and provides space for further heuristic optimisations like noise reduction. In the next step, this approach has been implemented in the presented `WorkloadClassificationAndForecasting` (WCF) system that enables online application and processes continuous forecast results of a variable number of different workload intensity behaviors.

Finally, the introduced approach has been evaluated conceptually by conducting experiments with the WCF system using real-world workload intensity traces as input. For example, in the first presented experiment, the relative error of the forecast points in relation to the arriving observations is reduced by 63% in average compared to the results of a static application of the Extended Exponential Smoothing (ETS) strategy that is by itself a sophisticated method and withstands direct comparisons to the other strategies used by the WCF system. The benefit of this approach has been demonstrated in a case study, showing that between 52% and 70% of the violations of a given service level agreement are prevented by applying proactive resource provisioning based on the forecast results of the introduced WCF system.

7.1 Future Work

The conduction of experiments to further analyse the trend interpolation strategies of the overhead group 2 with workload intensity traces that show less strong seasonal patterns and stronger trends is left for future work. Appropriate data for these experiments with strong trends has not been available, but is likely to be found in data of high resolution (seconds, minutes) or long term aggregated data (months, years).

The functionality of the WCF system could be further extended at its interfaces. One example for a possible extension is the combination of the WCF system with an intelligent filter that helps a system user to fit the aggregation level of the monitored arrival rates for specific forecast objectives like continuous short term forecast for proactive resource allocation or manually triggered long term forecasts for server capacity planning. For divergent forecast objectives, such a filter would multiplex a single input stream of monitoring data and provide the it as single workload intensity behaviors in different aggregation levels at the requires interface of the WCF system. A second example for a WCF system extension would be a combination with an anomaly detection system like ΘPAD as outlined in [Bie12] at the provides interface of the WCF system. Such a system would compute continuous anomaly ratings by comparing the WCF system's forecast results with the monitored data. The anomaly rating can serve to analyse the workload intensity behavior for sudden and unforeseen changes and in addition as an reliability indicator of the WCF system's recent forecast results that can easily be interpreted by a proactive resource manager.

It is planned to connect the WCF system's requires interface (data input) to a monitoring framework like Kieker [vHWH12] and provide the online forecast results to a performance model simulator that enables self-adaptive resource allocations as described in [HBK11]. A successful integration would enable more extensive experiments to validate the concepts of self-adaptive resource allocations based on workload forecasts of the WCF system for example by a comparison of using only a reactive provisioning method against the proactive forecasting approach in addition to the reactive. This way, the number of monitored SLA violations is directly comparable for equivalent workloads and system resources that run with the same start configuration. These integrated systems can also be used to dynamically reconfigure a virtualized cluster environment that executes a real-world application like for example SPEC's jEnterprise2010 benchmark [Sta09] and analyse the effects of the reconfigurations. The workload driver of this benchmark could be configured to execute a predefined workload intensity trace for the different request classes and not a constant or stepwise scaled workload intensity.

Acknowledgements

I would like to thank all people that supported me and made this diploma thesis' research work possible.

Particularly, I would like to thank Nikolaus Huber for his outstanding supervision - endless ideas, energy, countless hours of discussions and detailed, constructive feedback.

My thanks go to the IBM Research and Development Lab in Böblingen for making this intense and flexible cooperation possible and especially to Erich Amrehn - I appreciate the valuable discussions in numerous meetings - and Robert Vaupel for providing a number of real-world workload intensity traces.

Research and implementation results have been exchanged with the Christian-Albrechts University of Kiel with due to a related diploma thesis [Bie12]. I would like to thank the author Tillmann Carlos Bielefeld and his supervisor André van Hoorn for making a vivid exchange possible and especially for providing their implementation of the TSLIB as described in Section 4.4.1.

Bibliography

[ACCM09] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori, "Dynamic load management of virtual machines in cloud architectures," *Management*, pp. 201–214, 2009. [Online]. Available: http://samba.ing.unimo.it/papers/cloudcomp09.pdf

[ACG12] A. Amin, A. Colman, and L. Grunske, "An Approach to Forecasting QoS Attributes of Web Services Based on ARIMA and GARCH Models," in *Proceedings of the 19th IEEE International Conference on Web Services (ICWS)*. IEEE, 2012.

[AGC12] A. Amin, L. Grunske, and A. Colman, "An Automated Approach to Forecasting QoS Attributes Based on Linear and Non-linear Time Series Modeling," in *(Accepted for publication to appear) Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE/ACM, 2012.

[ALSS95] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim, "Fast similarity search in the presence of noise, scaling, and translation in time-series databases," in *Proceedings of the 21th International Conference on Very Large Data Bases*, ser. VLDB '95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 490–501. [Online]. Available: http://dl.acm.org/citation.cfm?id=645921.673155

[AMM+09] H. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, and D. Kaminsky, "Analysis of energy efficiency in clouds," in *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World:*, nov. 2009, pp. 416–421.

[AW96] M. F. Arlitt and C. L. Williamson, "Web server workload characterization: the search for invariants," *SIGMETRICS Perform. Eval. Rev.*, vol. 24, pp. 126–137, May 1996.

[BBR+07] M. Bensch, D. Brugger, W. Rosenstiel, M. Bogdan, W. G. Spruth, and P. Baeuerle, "Self-learning prediction system for optimisation of workload management in a mainframe operating system," in *ICEIS (2)*, 2007, pp. 212–218.

[Bie12] T. C. Bielefeld, "Online Performance Anomaly Detection for Large-Scale Software Systems," March 2012, Diploma Thesis, University of Kiel.

[BJR08] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis : forecasting and control*, 4th ed., ser. Wiley series in probability and statistics. Hoboken, NJ: Wiley, 2008, includes index.

[BKB07] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM*

'07. 10th IFIP/IEEE International Symposium on, 21 2007-yearly 25 2007, pp. 119 –128.

[BM04] M. N. Bennani and D. A. Menasce, "Assessing the robustness of self-managing computer systems under highly variable workloads," in *Proceedings of the First International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 62–69. [Online]. Available: http://dl.acm.org/citation.cfm?id=1078026.1078409

[CDM11] E. Caron, F. Desprez, and A. Muresan, "Pattern matching based forecast of non-periodic repetitive behavior for cloud clients," *J. Grid Comput.*, vol. 9, pp. 49–64, March 2011. [Online]. Available: http://dx.doi.org/10.1007/s10723-010-9178-4

[CEM+10] C. Chapman, W. Emmerich, F. G. Márquez, S. Clayman, and A. Galis, "Software architecture definition for on-demand cloud provisioning," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 61–72. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851485

[CS85] M. Calzarossa and G. Serazzi, "A characterization of the variation in time of workload arrival patterns," *IEEE Trans. Comput.*, vol. 34, pp. 156–162, February 1985. [Online]. Available: http://dl.acm.org/citation.cfm?id=1309285.1309718

[DLHS11] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011. [Online]. Available: http://pubs.amstat.org/doi/abs/10.1198/jasa.2011.tm09771

[Fro11] T. Frotscher, "Prognoseverfahren für das Antwortzeitverhalten von Software-Komponenten," March 2011, Christian-Albrechts-Universität zu Kiel, Bachelor's Thesis.

[GCP05] M. Goldszmidt, I. Cohen, and R. Powers, "Short term performance forecasting in enterprise systems," in *In ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. ACM Press, 2005, pp. 801–807.

[GKO+08] P. G. Gould, A. B. Koehler, J. K. Ord, R. D. Snyder, R. J. Hyndman, and F. Vahid-Araghi, "Forecasting time series with multiple seasonal patterns," *European Journal of Operational Research*, vol. 191, no. 1, pp. 207 – 222, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221707008740

[Goo10] P. Goodwin, "The holt-winters approach to exponential smoothing: 50 years old and going strong," *FORESIGHT*, vol. 19, 2010.

[GRCK07] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, ser. IISWC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 171–180. [Online]. Available: http://dx.doi.org/10.1109/IISWC.2007.4362193

[HB10] G. Herbst and S. F. Bocklisch, "Short-time prediction based on recognition of fuzzy time series patterns," in *Proceedings of the Computational intelligence for knowledge-based systems design, and 13th international conference on*

Information processing and management of uncertainty, ser. IPMU'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 320–329. [Online]. Available: http://dl.acm.org/citation.cfm?id=1876326.1876366

[HBK11] N. Huber, F. Brosig, and S. Kounev, "Model-based Self-Adaptive Resource Allocation in Virtualized Environments," in *6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011), Honolulu, HI, USA*, May 23-24 2011, acceptance Rate (Full Paper): 27% (21/76).

[HK06] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, pp. 679–688, 2006.

[HK08] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for r," pp. 1–22, 7 2008. [Online]. Available: http://www.jstatsoft.org/v27/i03

[HKPB02] R. J. Hyndman, M. L. King, I. Pitrun, and B. Billah, "Local linear forecasts using cubic smoothing splines," Monash University, Department of Econometrics and Business Statistics, Monash Econometrics and Business Statistics Working Papers 10/02, 2002. [Online]. Available: http://EconPapers.repec.org/RePEc:msh:ebswps:2002-10

[HMBN10] M. Hedwig, S. Malkowski, C. Bodenstein, and D. Neumann, "Towards autonomic cost-aware allocation of cloud resources," in *Proceedings of the International Conference on Information Systems ICIS 2010*, 2010. [Online]. Available: http://aisel.aisnet.org/icis2010_submissions/180/

[Hyn08] A. . O. K. . S. R. Hyndman, Rob ; Köhler, Ed., *Forecasting with Exponential Smoothing : The State Space Approach*, ser. Springer Series in Statistics. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008, in: Springer-Online. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71918-2

[JHJ+10] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, ser. ICDCS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 62–73. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2010.88

[KBHR10] S. Kounev, F. Brosig, N. Huber, and R. Reussner, "Towards self-aware performance and resource management in modern service-oriented systems," in *Proceedings of the 2010 IEEE International Conference on Services Computing*, ser. SCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 621–624. [Online]. Available: http://dx.doi.org/10.1109/SCC.2010.94

[KHvKR11] M. Kuperberg, N. R. Herbst, J. G. von Kistowski, and R. Reussner, "Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms," Informatics Innovation Center at Karlsruhe Institute of Technology, Karlsruhe, Germany, Tech. Rep., June 2011.

[KKK10] H. Kim, W. Kim, and Y. Kim, "Predictable cloud provisioning using analysis of user resource usage patterns in virtualized environment," in *Grid and Distributed Computing, Control and Automation*, ser. Communications in Computer and Information Science, T.-h. Kim, S. S. Yau, O. Gervasi, B.-H. Kang, A. Stoica, and D. Slezak, Eds. Springer Berlin Heidelberg, 2010, vol. 121, pp. 84–94.

[Kle] S. D. Kleeberg, “Neuronale Netze und Maschinelles Lernen zur Lastvorhersage in z/OS,” Universität Tübingen, Diploma Thesis.

[Kou05] S. Kounev, *Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction.* Shaker Verlag, Ph.D. Thesis, Technische Universität Darmstadt, Germany, Dec. 2005, best Dissertation Award from the ”Vereinigung von Freunden der Technischen Universität zu Darmstadt e.V.”. [Online]. Available: http://www.amazon.de/exec/obidos/ASIN/3832247130/302-7474121-6584807

[Kou07] S. Kounev, *Software Performance Evaluation.* John Wiley and Sons Incorporation, 2007. [Online]. Available: http://dx.doi.org/10.1002/9780470050118.ecse390

[LO10] L. Lefèvre and A.-C. Orgerie, “Designing and evaluating an energy efficient cloud,” *The Journal of Supercomputing*, vol. 51, pp. 352–373, 2010, 10.1007/s11227-010-0414-2. [Online]. Available: http://dx.doi.org/10.1007/s11227-010-0414-2

[MA98] D. A. Menascé and V. A. F. Almeida, *Capacity planning for Web performance: metrics, models, and methods.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[MHL+11] S. J. Malkowski, M. Hedwig, J. Li, C. Pu, and D. Neumann, “Automated control for elastic n-tier workloads based on empirical modeling,” in *Proceedings of the 8th ACM international conference on Autonomic computing*, ser. ICAC ’11. New York, NY, USA: ACM, 2011, pp. 131–140. [Online]. Available: http://doi.acm.org/10.1145/1998582.1998604

[MIK+10] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, “Efficient resource provisioning in compute clouds via vm multiplexing,” in *Proceedings of the 7th international conference on Autonomic computing*, ser. ICAC ’10. New York, NY, USA: ACM, 2010, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/1809049.1809052

[Mit09] T. Mitsa, *Temporal Data Mining*, ser. Chapman & Hall/CRC data mining and knowledge discovery series. Chapman & Hall/CRC, 2009. [Online]. Available: http://books.google.com/books?id=4P_7ydvW7cAC

[OAL11] A.-C. Orgerie, M. D. Assunção, and L. Lefèvre, “Energy aware clouds,” in *Grids, Clouds and Virtualization*, ser. Computer Communications and Networks, M. Cafaro and G. Aloisio, Eds. Springer London, 2011, pp. 143–166, 10.1007/978-0-85729-049-67. [Online]. Available: http://dx.doi.org/10.1007/978-0-85729-049-67

[OLG10] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, “Demystifying energy consumption in grids and clouds,” in *Green Computing Conference, 2010 International*, aug. 2010, pp. 335–342.

[Ols01] R. T. Olszewski, “Generalized feature extraction for structural pattern recognition in time-series data,” Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 2001, aAI3040489.

[Raa93] K. E. E. Raatikainen, “Cluster analysis and workload classification,” *SIGMETRICS Perform. Eval. Rev.*, vol. 20, pp. 24–30, May 1993. [Online]. Available: http://doi.acm.org/10.1145/155775.155781

[RBH+07] R. Reussner, S. Becker, J. Happe, H. Koziolek, K. Krogmann, and M. Kuperberg, *The Palladio component model*. Karlsruhe Institute of Technology, 2007.

[SGL+11] S. Spiegel, J. Gaebler, A. Lommatzsch, E. De Luca, and S. Albayrak, "Pattern recognition and classification for multivariate time series," in *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, ser. SensorKDD '11. New York, NY, USA: ACM, 2011, pp. 34–42. [Online]. Available: http://doi.acm.org/10.1145/2003653.2003657

[SH05] L. Shenstone and R. J. Hyndman, "Stochastic models underlying croston's method for intermittent demand forecasting," *Journal of Forecasting*, vol. 24, no. 6, pp. 389–402, 2005. [Online]. Available: http://dx.doi.org/10.1002/for.963

[Shu11] R. H. Shumway, *Time Series Analysis and Its Applications : With R Examples*, ser. Springer Texts in StatisticsSpringerLink : Bücher, D. S. Stoffer, Ed. New York, NY: Springer Science+Business Media, LLC, 2011. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-7865-3

[Sta09] *SPEC jEnterprise 2010 User Guide*, Standard Performance Evaluation Cooperation, 2009. [Online]. Available: http://www.spec.org/jEnterprise2010/docs/UsersGuide.html

[SWHB06] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Open versus closed: a cautionary tale," in *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, ser. NSDI'06. Berkeley, CA, USA: USENIX Association, 2006, pp. 18–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267680.1267698

[TBL09] R. Talaber, T. Brey, and L. Lamers, "Using virtualization to improve data center efficiency," The Green Grid, Tech. Rep., 2009.

[TL10] J. Temple and R. Lebsack, "Fit for Purpose: Workload Based Platform Selection," IBM Corporation, Tech. Rep., 2010.

[Urb11] S. Urbanek, *Rserve: Binary R server*, 2011, r package version 0.6-5. [Online]. Available: http://CRAN.R-project.org/package=Rserve

[vHRH08] A. van Hoorn, M. Rohr, and W. Hasselbring, "Generating probabilistic and intensity-varying workload for web-based software systems," in *Proceedings of the SPEC international workshop on Performance Evaluation: Metrics, Models and Benchmarks*, ser. SIPEW '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 124–143.

[vHWH12] A. van Hoorn, J. Waller, and W. Hasselbring, "Kieker: A framework for application performance monitoring and dynamic software analysis," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*. ACM, Apr. 2012, invited tool demo paper. To appear.

[VHZC10] J. Verbesselt, R. Hyndman, A. Zeileis, and D. Culvenor, "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series," *Remote Sensing of Environment*, vol. 114, no. 12, pp. 2970 – 2980, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425710002336

[vMvHH11] R. von Massow, A. van Hoorn, and W. Hasselbring, "Performance simulation of runtime reconfigurable component-based software architectures," in

5th European Conference on Software Architecture (ECSA '11), ser. Lecture Notes in Computer Science, I. Crnkovic, V. Gruhn, and M. Book, Eds., vol. 6903. Springer, Sep. 2011, pp. 43–58.

[WL05] T. Warren Liao, "Clustering of time series data-a survey," *Pattern Recogn.*, vol. 38, pp. 1857–1874, November 2005. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2005.01.025

[ZLDL11] X. Zhang, J. Liu, Y. Du, and T. Lv, "A novel clustering method on time series data," *Expert Syst. Appl.*, vol. 38, pp. 11 891–11 900, September 2011. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2011.03.081

List of Tables

List of Figures

Glossary

application scalability is a property of the software/application layers. The application is able to maintain its performance goals in terms of response time or throughput as defined in Service Level Agreements (SLA) even when its workload intensity increases. Platform scalability is a necessary property to make application scalability possible.. 9, 87

elasticity of a computing system is characterized by the temporal and quantitative properties of automated scaling, which is run-time resource provisioning and unprovisioning performed by the execution platform. A manually scaled system cannot be called elastic. Execution platform elasticity depends on the state of the platform and on the state of the platform-hosted applications. Elasticity implies scalability beneath a given upper bound [KHvKR11]. 10, 69, 87

execution platform consists of hardware, virtualization and operating system layer plus optional middleware like an application server. 9, 10, 87, 88

platform scalability is the ability of the execution platform to provide and make use of as many (additional) resources as needed (or explicitly requested) by an application.. 9, 87

request is submitted to a software service by a user and can be seen as an encapsulation of a single usage of a service. 10, 11, 23, 24, 87, 88

request class is a category of requests that is characterized by statistically indistinguishable resource demands. 10, 23, 25, 43, 88

resource demand in units of time or capacity is the consumption of physical or virtual resources induced by processing a single request. 10, 23, 24, 87

resource utilization is the fraction of time that the resource is busy. [Kou05]. 10, 11, 24

response time is the time it takes a system to react to a request and is composed of congestion time(s) and service time(s). [Kou05]. 9–11, 24, 64, 70, 87

scalability A computing system consisting of an execution platform and applications is scalable if both platform scalability and application scalability are fulfilled. These properties imply the absence of bottlenecks in both active and passive resources. Scalability of a computing system can be limited by an upper bound of workload intensity or resource pool size. [KHvKR11]. 9, 10, 87

software service is offered by a computing system to the users of the system, which can be human persons via an interface or computing machines. In our context, a software service can be seen as a deployed software component. 10, 87

throughput is the rate at which requests are completed by a computer system (measured in operations per unit of time). [Kou05]. 9–11, 24, 87

time series X is a discrete function that represents real-valued measurements $x_i \in R$ for every time point t_i in a set of n equidistant time points $t = t_1, t_2, ..., t_n$: $X = x_1, x_2, ..., x_n$ as described in [Mit09]. A time series may have a finite capacity. The time between to time series points is defined by a value and a time unit. The number of time series points, that add up to a higher time unit or another obvious period is the frequency of a time series. The frequency attribute of a time series is an important start value for the search of seasonal patterns.. 11–13, 23–25, 27–29, 88

time series of request arrival rates is a time series that contains $n_i \in N$ sums of unique request arrivals during the corresponding time interval $[t_i, t_{i+1})$. 11, 23, 25–28, 88

usage scenario is an instance of a workload model that defines the rates and order of service requests and is used for early performance predictions in the context of the Palladio Component Model (PCM) as described in [RBH+07]. 10

virtualized elastic system is an execution platform that is elastic in terms of the given definition and makes use of virtualization technologies. By using virtualization technology the underlying physical resources can be mapped transparently and dynamically to virtual resources. 10

workload is the physical (not modeled) usage of a system over time containing requests of one or more request classes. A workload can contain usage patterns that enable load forecasting, provisioning, planning or anomaly analysis. This definition deviates from the definition in [TL10] on page 2, where a workload is a more general term capturing applications and their service level agreements additionally. 10, 88

workload category is a coarse-grained group of workloads and divided into four basic application and technology domains broadly used for market segmentation and analysis. The four categories are:

- Database and Transaction Processing
- Business Process Applications
- Analytics and High Performance Computing
- Web Collaboration and Infrastructure

as in [TL10] on page 13. 10

workload intensity behavior is a description of a workload's characteristic changes in intensity over time like seasonal patterns, trends, noise, burstiness, level and more like positivity or burstiness. The workload intensity behavior can be extracted from a corresponding time series of request arrival rates. 11, 13, 15, 16, 23–25, 27–30, 33, 35–37, 43, 47–49, 52, 55, 56, 64, 67, 68, 73, 88

workload model is a representation that captures the main aspects of the real workload that has effect on the performance measures of interest. [Kou07]. 10, 11, 88