CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2014; **00**:2–28
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning

Nikolas Roman Herbst[1]*, Nikolaus Huber[1], Samuel Kounev[1], and Erich Amrehn[2]

[1]*Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131 Karlsruhe, Germany*
[2]*IBM Research & Development, Schoenaicher Str. 220, 71032 Boeblingen, Germany*

SUMMARY

As modern enterprise software systems become increasingly dynamic, workload forecasting techniques are gaining in importance as a foundation for online capacity planning and resource management. Time series analysis offers a broad spectrum of methods to calculate workload forecasts based on history monitoring data. Related work in the field of workload forecasting mostly concentrates on evaluating specific methods and their individual optimisation potential or on predicting Quality-of-Service (QoS) metrics directly. As a basis, we present a survey on established forecasting methods of the time series analysis concerning their benefits and drawbacks and group them according to their computational overheads. In this paper, we propose a novel self-adaptive approach that selects suitable forecasting methods for a given context based on a decision tree and direct feedback cycles together with a corresponding implementation. The user needs to provide only his general forecasting objectives. In several experiments and case studies based on real-world workload traces, we show that our implementation of the approach provides continuous and reliable forecast results at run-time. The results of this extensive evaluation show that the relative error of the individual forecast points is significantly reduced compared to statically applied forecasting methods, e.g. in an exemplary scenario on average by 37%. In a case study, between 55% and 75% of the violations of a given service level objective can be prevented by applying proactive resource provisioning based on the forecast results of our implementation. Copyright © 2014 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Workload Forecasting, Arrival Rate, Time Series Analysis, Proactive Resource Provisioning, Assurance of Service Level Objectives

1. INTRODUCTION

Virtualization allows to dynamically assign and release resources to and from hosted applications at run-time. The amount of resources consumed by the executed software services are mainly determined by the current workload intensity. They both typically vary over time according to the user's behavior. The flexibility that virtualization enables in resource provisioning and the variation of the resource demands over time 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 as discussed for example in [1, 2].

Commonly, mechanisms that try to continuously match the amounts of demanded to provisioned resources are reactive as they use threshold based rules to detect and react on resource shortages. However, such reactive mechanisms can be combined with proactive ones that anticipate changes in

*Correspondence to: Nikolas Roman Herbst, Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131 Karlsruhe, Germany, E-mail: herbst@kit.edu resource demands and proactively reconfigure the system to avoid resource shortages. For building such proactive mechanisms, the time series analysis offers a spectrum of sophisticated forecasting methods. But as none of these methods is the overall best performing one, we work on the idea to intelligently combine the benefits of the individual forecasting methods to achieve higher forecast accuracy independent of the forecasting context.

Related research in the field of proactive resource provisioning as it can be found in [3, 4, 5, 6, 7, 8] mostly concentrates on single forecasting methods of the time series analysis and their individual optimisation potential. This way, reliable forecast results are achieved only in certain situations. In addition, the forecasting methods are applied in the majority of cases on monitored QoS metrics that depend on both the recent workload intensity and the system state. Therefore, these forecast results do not allow to directly estimate the amount of arriving work for near-future online resource provisioning.

In this paper, we present a survey of forecasting methods offered by the time series analysis that base their computations on periodically monitored arrival rate statistics. We group the individual forecasting methods according to their computational overhead, their benefits and drawbacks as well as their underlying assumptions. Based on this analysis, we design a novel forecasting methodology that dynamically selects at run-time a suitable forecasting method for a given context. This selection is based on a decision tree that captures the users' forecasting objectives, requirements of individual forecasting methods and integrates direct feedback cycles on the forecast accuracy as well as heuristics for further optimisations using data analysis techniques. Users need to provide only their general forecasting objectives concerning forecasting frequency, forecast horizon, accuracy requirements and overhead limitations and are not responsible to dynamically select appropriate forecasting methods. Our implementation of the proposed Workload Classification and Forecasting (WCF) approach is able to continuously provide time series of point forecasts of the workload intensity with confidence intervals and forecast accuracy metrics in configurable intervals and with controllable computational overhead during run-time.

In summary, the contributions of this paper are: (i) We identify the characteristics and components of an observed workload intensity behavior and define metrics to quantify these characteristics for the purpose of automatic classification and selection of a forecasting method. (ii) We provide a survey of state-of-the-art forecasting approaches based on time series analysis (including interpolation, decomposition and pattern identification techniques) focusing on their benefits, drawbacks and underlying assumptions as well as their computational overheads. (iii) We propose an novel forecasting methodology that self-adaptively correlates workload intensity behavior classes and existing time series based forecasting approaches. (iv) We provide an implementation of the proposed WCF approach as part of a new `WorkloadClassificationAndForecasting` (WCF) system[†] specifically designed for run-time usage. (v) We evaluate our WCF approach using our implementation in the context of multiple different experiments and case studies based on real-world workload intensity traces.

The results of our extensive experimental evaluation considering multiple different scenarios show that the dynamic selection of a suitable forecast method significantly reduces the relative error of the workload forecasts compared to the results of statically selected fixed forecasting methods, e.g. in the presented exemplary scenario on average by 37%. Even when limiting the set of forecasting methods considered in the dynamic selection, our self-adaptive approach achieves a higher forecast accuracy with less outliers than individual methods achieve in isolation. In the presented case study, we apply rule-based proactive resource provisioning interpreting the forecast results of the introduced WCF approach. This way, we manage to prevent between 55% and 75% of the violations of a given service level objective (SLO).

The remainder of the paper is structured as follows: In Section 2, we define several concepts and terms that are crucial for understanding the presented approach. In the next step, we analyze characteristics of a workload intensity behavior (*WIB*) and give a compact survey on existing workload forecasting methods based on time series analysis. We present our self-adaptive approach

[†]http://www.descartes-research.net/tools/

for workload classification and forecasting (WCF) in Section 3. Section 4 shortly summarizes the component-based architecture of the WCF implementation. In Section 5, we evaluate the WCF approach by presenting and discussing an exemplary experiment and a case study. We review related work in Section 6, and present some concluding remarks in Section 7.

2. FOUNDATIONS

In this section, we start by defining crucial terms in the context of software services, workload characterization and performance analysis that are helpful to build a precise understanding of the presented concepts and ideas and are used in the rest of the paper:

We assume that *software services* are offered by a computing system to a set of users which can be humans or other systems. In our context, a software service can be seen as a deployed software component.

Each *request* submitted to the software service by a user encapsulates an individual usage of the service.

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

A *resource demand* in units of time or capacity is a measure of the consumption of physical or virtual resources incurred for processing an individual request.

The term *workload* refers to the physical usage of the system over time comprising requests of one or more request classes. This definition deviates from the definition in [9], where *workload* is defined as a more general term capturing in addition to the above the respective applications and their SLOs.

A *workload category* is a coarse-grained classification of a workload type with respect to four basic application and technology domains as defined in [9]: (i) Database and Transaction Processing, (ii) Business Process Applications, (iii) Analytics and High Performance Computing, (iv) Web Collaboration and Infrastructures.

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 [10]. The elapsed time between two points in the time series is defined by a value and a time unit.

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

A *workload intensity behavior* (*WIB*) is an abstract description of a workload's characteristic changes in intensity over time describing the shape of seasonal patterns and trends as well as the level of noise and bursts as further described in the following section. The *WIB* can be extracted from a corresponding time series of request arrival rates.

2.1. Workload Intensity Behavior

In Paper [11] from 1985, the authors use polynomial regression to identify the variation in time of workload arrival patterns and describe typical daily workload patterns. The search for invariants in web-server workloads focusing on characteristics of request classes and their typical resource demands is covered in Paper [12] from 1996. In [13], the authors discuss and evaluate the two basic ways to model different workload types either in an open or a closed workload model. Modeling intensity variations of workloads for generating realistic benchmarking scenarios is addressed in [14]. In Report [15], the research focuses on establishing a relationship between workload intensity characteristics and a suitable elasticity controlling mechanisms.

The above listed research work on workload classification and modeling identifies central characteristics of workload intensity behaviors (*WIBs*) in general, but the focus is not to establish any connection between a certain characteristic and a suitable forecasting method. As *WIBs* and the corresponding time series of request arrival rates are closely connected, the components of the time series itself and its properties are central to our further analysis.

According the theory of time series analysis [16, 17, 18], a time series can be decomposed in an additive or multiplicative manner into the following three components, whose relative weights and shapes characterise the respective *WIB*:

The *trend* component can be described by a monotonically 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, it cannot be forecast based on historic observations but detected retrospectively. It is possible to estimate the likelihood of a change in the trend component by analysing the durations of historic trends.

The *season* component captures recurring patterns that are composed of one or more frequencies, e.g. daily, weekly or monthly patterns. These predominant 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. The noise can be reduced by applying smoothing techniques like weighted moving averages (WMA), by using lower sampling frequency, or by a low-pass filter that eliminates high frequencies. Finding a suitable trade-off between the amount of noise reduction and the respective potential loss of information can enhance forecast accuracy.

A time series decomposition into the components mentioned above is illustrated in Figure 1. This decomposition of a time series has been presented in [19]. The authors offer an implementation of their approach for time series decomposition and detection of breaks in trends or seasonal components (BFAST)[‡]. In the first row, the time series input data is plotted. The second row contains detected (yearly) seasonal patterns, whereas the third row shows estimated trends and several breaks within these trends. The remainder in the undermost row is the non-deterministic noise component computed by the difference between the original time series data and the sum (or product) of the trend and the seasonal components.

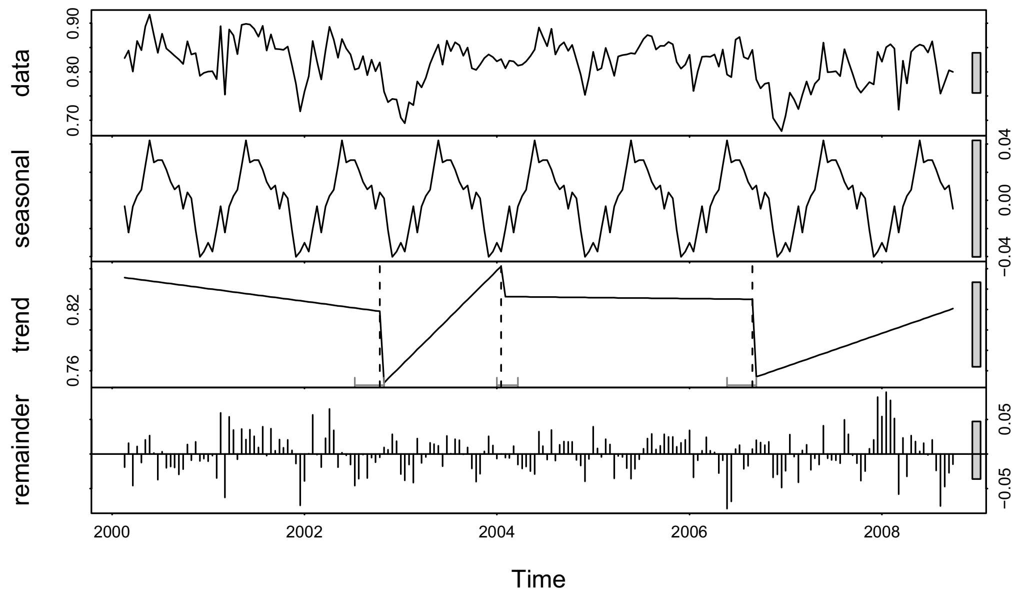


Figure 1. Time Series Decomposition into Season, Trend and Noise Components as presented in [19]

A nearly complete decomposition of a time series into its three main components as described above by using the BFAST approach [19] induces a high computational burden. Hence, fast computable measures that characterise a time series are crucial for efficient workload classification

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

at run-time. Table I summarizes the following set of measures, we apply for online *WIB* characterisation and classification:

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

The *length* of the time series mainly influences the accuracy of approximations for the components mentioned above and limits the space of applicable forecasting methods.

The *relative monotonicity* is maximum number of consecutive monotonic values either upwards or downwards within a sliding window related to the length of the sliding window and indirectly characterises the influence of the noise and seasonal components. A respective small value can be seen as a sign of a high noise level and therefore as a hint to apply a time series smoothing technique.

The *maximum*, the *median* and the *quartiles* are important indicators for the spread of the distribution of the time series data and can be unified in the *quartile dispersion coefficient* (QDC) defined as the distance of the quartiles divided by the median value.

The standard deviation and the mean value are combined in the *coefficient of variation* (COV) which also characterizes the spread of the distribution as a dimensionless quantity.

Absolute *positivity* of a time series is another important characteristic. Numeric stability of several forecasting methods is not given if a time series contains zero values. As arrival rates are per se non-negative values, a time series that is not absolutely positive should be subjected to a simple filter that eliminates the negative values or analyzed using specialized forecasting method.

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

We define a *relative gradient* as the absolute gradient of the latest quarter of a time series period minus the median of this quarter. It captures the steepness of the latest quarter period as a dimensionless value. 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 pattern or burst. Simple trend interpolation methods tend to over- or underestimate the development in these steep intervals.

The presented metrics are sufficient to capture the most important characteristics of a *WIB*, though they have low computational complexity. Hence, they are of major importance for our online workload classification process.

2.1.1. Real-World WIBs: Commonly, human users trigger software services either directly in an interactive manner or indirectly (e.g. scheduled batch tasks). Thus, real-world *WIB* traces are likely to show strong recurring patterns in daily periods possibly overlaid by far longer seasonal periods of weeks, months or years. The typical shape of a daily seasonal pattern characterises the *WIB* of a software service. These patterns are directly influenced by common human habits like working hours, lunch time and common sleeping hours. A calendar that defines working days, weekends and holidays may also have an strong impact on the *WIB*. As a variety of factors, e.g. the weather, influences human usage behavior, real-world *WIBs* usually contain unpredictable parts.

A high weight of a trend component within a *WIB* trace is rarely seen at the scale of hours and days and more likely to be found in aggregated data at the scale of weeks, month or years. Therefore, the trend component is of major importance for long term forecasts. At the scale of seconds, minutes and hours sudden bursts in the *WIB* (either induced by planned events or anomalies) can have a strong impact by overlaying the normal daily pattern.

2.2. *Survey of Forecasting Methods*

In this section, we compare the most common forecasting approaches based on the time series analysis and highlight their requirements, advantages and disadvantages presenting a short summary based on [16, 17, 20, 18, 21, 22, 23]. All considered forecasting methods have been implemented

Table I. Means for Characterisation and Classification of Workload Intensity Behaviors

Name	Calculation	Expressiveness
Burstiness Index	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.
Relative Monotonicity	By iteration through the time series values the number of consecutive monotonic values (up or downwards) is counted. The *RelativeMonotonicity* is the maximum of these related to the *Length* of the analyzed time series span.	The *RelativeMonotonicity* 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.
Quartile Dispersion 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)
Relative Gradient	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.
Standard Deviation	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.
Weighted Moving Averages (MA)	Weighted *MovingAverages* (MA) 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)).	Weighted *MovingAverages* 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.

Table II. Forecasting Methods based in Time Series Analysis

Name	Operating Mode	Horizon	Requirements	Overhead	Strengths	Weaknesses	Optimal Scenario
Naïve Forecast ≅ ARIMA 100	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) ≅ ARIMA 001	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) ≅ ARIMA 011	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 , no interpolation	time series with some noise and changes within trend, but no seasonal behavior
Cubic Smoothing Splines (CS) [21] ≅ ARIMA 022	Cubic splines are fitted to the one-dimensional 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** (no improved 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 using SES [22]	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 (no 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 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 [20]	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 enables identification of complex seasonal time series patterns by FFT for 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

by Hyndman in the R forecast package[§] and documented in [20]. The implemented forecasting methods are based either on the state-space approach [17] or the ARIMA (auto-regressive integrated moving averages) approach for stochastic process modeling [16]. These two general approaches for stochastic process modeling have common aspects, but are not identical as in both cases there exist model instances that have no counterpart in the other approach and both have different strength and weaknesses as discussed, e.g., in [20].

In Table II, we briefly describe the considered forecasting approaches which are ordered according to their computational complexity. The first two below listed forecasting methods are very basic ones, whereas the third to seventh method include the capability to estimate trends and can be considered as instances of the ARIMA or ETS stochastic process modeling frameworks not covering any modeling of seasonal patterns. The last three forecast methods listed are capable to additionally capture seasonal patterns in different peculiarity. These three stochastic process modeling frameworks come together with model estimation procedures as described in [20].

2.2.1. Naive Forecasting Method: The *naive* method has no inherent complexity as it is based on 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 normalized comparability with other forecasting methods. 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 *moving averages* method with a sliding window size of one.

2.2.2. 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.

2.2.3. 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. Experiments showed that the SES method returns a result below 80 milliseconds 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 seasonal 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 2.2.9.

2.2.4. Cubic Smoothing Splines: As demonstrated and discussed in [22], *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. This method is an instance of the ARIMA($(p, d, q) = (0, 2, 2)$) model as described in 2.2.9 with a restricted parameter set that does not impair the forecast accuracy. This approach tends to better estimate trends than the SES method, but still seasonal patterns are not captured. This method tends to overestimate trends 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.

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

2.2.5. 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 2.2.9 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 the Book [18]. This method is not as sensitive as the CS method to steep parts in a time series. As the SES and CS methods, the application of ARMA($(p, q) = (1, 1)$) includes 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.

2.2.6. Croston's Method for Intermittent Time Series: Croston's method is presented in [23] and is specialized for forecasting of intermittent time series. In contrast to the other methods, this method 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 afterwards. 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.

2.2.7. Extended Exponential Smoothing: As presented in [17] and [20], the *extended exponential smoothing* (ETS) bases on the innovation state space approach and can explicitly model 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. The forecasting process starts in the first step by selecting an optimized model instance, 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 presence of complex seasonal patterns, the ETS often fails to model those and returns quicker with a worse forecast accuracy compared to the following two methods.

2.2.8. tBATS Innovation State Space Modelling Framework: The *tBATS* innovation state space modelling framework has been presented in 2011 in [21] 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 resulted in more appropriate forecast accuracy compared to the ETS method.

2.2.9. ARIMA Stochastic Process Modelling Framework: The *ARIMA* (auto-regressive integrated moving averages) stochastic process modelling framework is extensively presented in the Book [16]. 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 [20] 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 characteristic itself and not only on the amount of data to base the forecast result on. Experiments also showed that this ARIMA approach in most cases achieves closer confidence intervals than the tBATS approach.

Besides from [20], detailed information on the implementation, parameters and model estimation of the individual methods can be taken from cited sources within Table II. Capturing the complexity in common O-notation in most cases is infeasible except for the two first simple cases. The shape of seasonal patterns contained in the data as well as the used optimisation thresholds during a model selection and fitting procedures strongly influence the computational costs. The length of time series size represents only one part of the problem description. Therefore, the computational complexity of the individual methods has been evaluated based on experiments with a representative amount of forecasting method executions on a machine with a Intel Core i7 CPU (2.7 GHz). The forecasting methods make use of only a single core as multi-threading is not yet fully supported by the existing implementations.

3. APPROACH FOR WORKLOAD CLASSIFICATION AND FORECASTING

In this section, we present a self-adaptive workload classification and forecasting (WCF) process which can be used for selecting suitable forecasting methods at run-time.

Over time, a *Workload Intensity Behavior* (*WIB*) may change and develop in a way that affects its characteristics, i.e., the class of the *WIB* is not fixed and needs to be updated periodically. Therefore, our classification process must be self-adaptive. Therefore, it must consider changes of the *WIB* and of given forecasting objectives to adapt the assignment of appropriate forecasting methods to given *WIBs*.

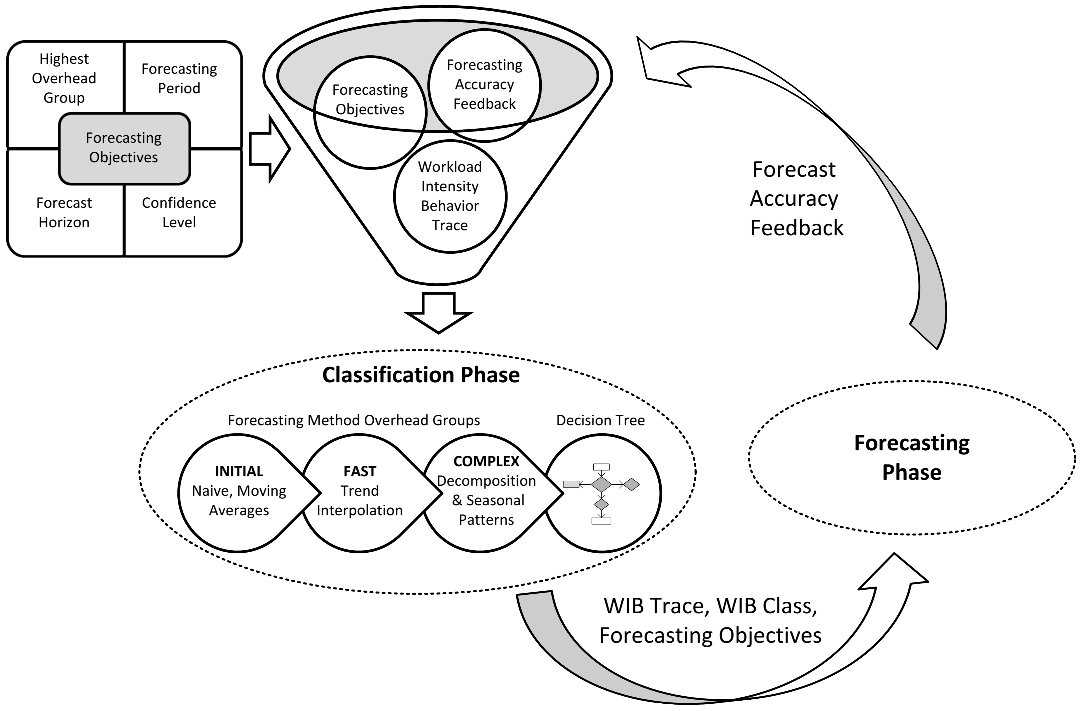

Highest Overhead Group
Forecasting Period
Forecasting Objectives
Forecast Horizon
Confidence Level
Forecasting Objectives
Forecasting Accuracy Feedback
Workload Intensity Behavior Trace
Forecast Accuracy Feedback
Classification Phase
Forecasting Method Overhead Groups
Decision Tree
INITIAL Naive, Moving Averages
FAST Trend Interpolation
COMPLEX Decomposition & Seasonal Patterns
Forecasting Phase
WIB Trace, WIB Class, Forecasting Objectives

Figure 2. Overview of the workload classification and forecasting (WCF) process.

An overview of the WCF process is sketched in Figure 2. Input of the WCF process are a trace of a *WIB*, a set of *forecasting objectives*, and possible feedback about the accuracy of the previous forecast. Essentially, we distinguish two phases in this process, the CLASSIFICATION phase and the FORECASTING phase. In the CLASSIFICATION phase, we extract the characteristics of a given *WIB* trace. Based on the identified characteristics, we use a decision tree (cf. 3) to classify the *Workload Intensity Behavior* and to select suitable forecasting methods. The assignment of forecasting methods also defines the class of the *WIB*. Then, in the FORECASTING phase, we apply the assigned forecasting method according to the given forecasting objectives. Our process also supports evaluating the accuracy of forecasts and uses the evaluation results as feedback for the next classification cycle. The classification and forecasting phases are executed iteratively, with a frequency that can be specified by the user. Thereby, our approach is able to continuously adapt the classification of the *WIB* based on its evolution. However, the two phases must not necessarily follow each other iteratively. They can also be executed in parallel, i.e., the classification of the *WIB* can be updated while it is also forecast. In the following, we explain in further details the individual parts of our self-adaptive WCF approach.

3.1. *Forecasting Objectives*

Time series forecasts can be used for a variety of purposes like short term proactive resource provisioning or long term capacity planning. Depending on the purpose, different forecasting methods with different configuration might provide better or quicker results. Our approach employs *forecasting objectives* which guide forecast result processing and control the overhead. Our approach supports specifying the following parameters as *forecasting objectives*:

1. The *Highest Overhead Group* parameter is a value of the interval $[1, 4]$ that specifies the highest overhead group the WCF approach can choose from. This value refers to the overhead groups we introduce in Subsection 3.2.
2. The *Forecast Horizon* parameter is a tuple of two positive integer values quantifying the number of time series points to be forecast. The first value of the tuple defines the *Start Horizon* and defines the number of time series points to be forecast at the beginning. The second value defines the *Maximum Horizon*, i.e., the maximum number of time series points to be forecast. The start value can then dynamically increased stepwise up to the maximum value. This is necessary because of the significant differences of the forecasting methods in terms of processing times and feasibility for long term forecasts.
3. The *Confidence Level* parameter can be a value $\alpha \in [0, 100)$ and defines the percentage of how many of the calculated confidence intervals have to include the forecast mean value.
4. The *Forecasting Period* parameter is a positive integer i specifying how often in terms of the number of time series points a forecasting method is executed. For example, if $i = 1$ a forecast is executed for every new time series point that is added to the time series. If $i = 10$, the forecast is executed every tenth time series point.

By specifying this parameters according to the characteristics of the considered forecasting scenario, one can influence the classification and forecasting process in finding a suitable forecasting method for the given *WIB*. For example, assume an offline capacity planning scenario, where there is a lot of time to analyze the *WIB* and forecast a relatively long time period. Then, one can set the objectives to the highest overhead group and to the desired maximum forecast horizon. Furthermore, if high forecasting confidence is required, one could also set $\alpha = 95$ to specify a high confidence level. Proposed settings and further explanations can be found in Table III.

3.2. *Forecasting Methods Overhead Groups*

In Section 2.2, we have presented a survey of different time series forecasting methods and discussed their computational overhead. In the following, we divide these forecasting methods into the following groups.

Table III. Forecasting Objectives Parameters

Parameter Name	Parameter Range	Proposed Configuration	Explanation
Forecasting Period	[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 result is requested every new time series point and can be dynamically increased by the *HorizonFactors* in the classification setting (Tab. 5) to reach the configured *MaximumHorizon*. 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 methods 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 methods 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 *StartHorizon* defines the number of time series points to be forecasted at the beginning and can be dynamically increased by HorizonFactors in the classification setting (Tab.5) 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* attribute of the time series, as a higher horizon setting may lead to broad confidence intervals.

Group 1 – Negligible Overhead
This group contains methods with almost no overhead, which are the *Moving Average (MA)* and the *Naive* forecasting methods.

Group 2 – Low Overhead
This group contains methods with low overhead and contains the fast forecasting methods *Simple Exponential Smoothing (SES)*, *Cubic Spline Interpolation (CS)*, the predefined *ARIMA101* model and the specialized *Croston's Method* for intermittent time series. The processing times of forecasting methods in this group are below 100 ms for a maximum of 100 time series points.

Group 3 – Medium Overhead
This group stands for medium overheads and contains the forecasting methods *Extended Exponential Smoothing (ETS)* and *tBATS*. The processing times are below 30 seconds for less than 200 time series points.

Group 4 – High Overhead
This group stands for high overheads and contains again *tBATS* and additionally the *ARIMA* forecasting framework with automatic selection of an optimal *ARIMA* model. The processing times for methods in this group are below 60 seconds for less than 200 time series points.

3.3. *Partitions of the Classification Process*

In addition to groups of computational overhead, we distinguish three major partitions of forecasting methods. Each partition contains forecasting methods which are applicable in different situations during run-time workload classification and forecasting process. When a time series of request arrival rates is added for classification and no historic data of its *WIB* is available yet, the only option is to apply fast and naive forecasting methods to get a basic forecast. At a later point in time, when more observations become available, it may be useful to apply methods 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.

Therefore, our classification process distinguishes three partitions according to the amount of historic data available in the time series: an *initial* partition, a *fast* partition, and a *complex* partition.

These partitions are related to the overhead groups of the forecasting methods. Having a short time series, i.e. for the initial partition, only forecasting methods in the *Overhead Group 1* can be applied. A medium length of a time series may allow application of methods contained in the *Overhead Group 2* and a long time series (complex partition) enables the use of the methods 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 setting of the classification process. Based on the experience gained from experiments, we recommend to set the threshold for the transition from initial to fast to a value $\tau \in [5, \frac{p}{2}]$ observations (five being the minimal amount of observations needed for Cubic Spline Interpolation and p being the length of a period in time series points). The fast to complex threshold should be set to a value $\tau \geq 3p$ because most methods in the respective overhead group need at least three pattern occurrences to identify them. A short summary of the capabilities of this three different classification strategies (initial, fast and complex) can be found in Table IV. How these capabilities are realised, is discussed in detail in the following sections.

Table IV. Classification Strategies and their Capabilities

Classification Strategy	Overhead Group	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.

3.4. *Evaluating Forecasting Accuracy*

In an online scenario, *WIBs* are not stationary, i.e., their characteristics can change over time. Accordingly, the most appropriate forecasting method can also change and must be adapted at run-time. It is the responsibility of the classification process to detect such changes and adjust the WIB class to the forecasting method that yields the most accurate results for the given *WIB*, considering given forecast objectives.

To evaluate the accuracy of a forecasting method, a number of metrics assessing the differences between forecast results and corresponding observations have been proposed (cf. [24] for an overview). As also extensively discussed by [24], the *Mean Absolute Scaled Error* (MASE) is the metric of choice that enables consistent comparisons of forecasting methods across different data samples. The MASE metric for an interval $[m, n] \in T$ is defined as follows:

$$MASE[m,n] = \frac{1}{n-m+1} \sum_{t=m}^{n} (\left| \frac{error_t}{b_{[m,n]}} \right|),$$

whereas $error_t$ is defined as

$$error_t = forecast_t - observation_t, \; t \in [m,n]$$

and $b_{[m,n]}$ as the average change within the observation

$$b_{[m,n]} = \frac{1}{n-m} \cdot \sum_{i=m+1}^{n} |observation_i - observation_{i-1}| \, .$$

Essentially, the MASE metric compares the forecast accuracy with the accuracy of the naive forecasting method. If the MASE value is close to one or even bigger, the computed forecast results are not valuable for further processing because their accuracy is equal or even worse than using the naive forecasting method. This means that one could use monitoring data for resource provisioning, directly. In turn, the closer the metric value is to zero, the better the accuracy of the selected forecasting method.

In our classification process, we select the forecasting method providing the lowest MASE value for the given *WIB*. The best point in time during the WCF process to precisely calculate the MASE metric is just before the next forecast execution is triggered, i.e., when both the observation values as well as the forecast results are available. This result is used as forecast accuracy feedback in the classification phase. However, in an online scenario where we need forecast results for proactive resource provisioning, no observations are available to assess the quality of the forecasts. In such situations when no observations are available, the MASE metric can also be used to assess the accuracy of different forecasting methods by comparing them with the naive forecast.

Thus, we execute two or more forecasting methods in parallel during the forecasting phase and compare their accuracy in configurable intervals using the MASE metric to ensure that the currently chosen WIB class still produces the most accurate results compared to other approaches. This comparison of forecasting method is also triggered when the evaluation of forecast results with observations seems implausible or shows low accuracy ($MASE[m, n] \geq 1$).

3.5. Non-Absolutely Positive Workloads

The workloads considered in this work contain time series of request arrival rates. These time series are non-negative, as there cannot be a negative number of requests per time period. However, the time series might contain zero values, as there might exist time periods where no requests arrive. The problem is that the majority of forecasting methods assume absolutely positive time series as input (cf. 2.2). Such forecasting methods are not numerically stable, i.e., they interrupt after a division by zero and thus cannot return a forecast result. Therefore, we need to eliminate zeros from the time series before passing it to the forecasting method. However, if zero values appear regularly in the time series, it is better to use Croston's forecasting method for intermittent demands which is developed for such time series. This method decomposes the time series into two different time series. A strictly positive time series and another time series containing the period duration when the time series was zero. The forecast is then executed independently for both time series and combined later on. The classification of a *WIB* would be highly sensitive to zero values if only a few observations of zero values immediately caused a switch to the Croston's forecast method. To configure this sensitivity, we introduced a threshold for the rate of zero values. We recommended set this threshold to a reasonable value between 20% and 40%.

3.6. Decision Tree

The decision tree depicted in Figure 3 is the core element for selecting a suitable forecasting method. To determine a suitable forecasting method for a given workload, one follows the branches by evaluating the identified *WIB* characteristics (cf. I). The thresholds in the conditions of the branches are parameterized. The values of these thresholds are either derived from the forecasting objectives or based on empirical values obtained during our experiments. The leaves of the decision tree contain one or several recommended forecasting methods. In case a leave contains more than one suitable forecasting method, the forecasting phase executes all of them and evaluates their estimated forecast accuracy using the MASE metric to select the more accurate result. Before the next iteration of the forecasting phase, i.e., when real system observations are available, we can also determine the accuracy of the forecast values compared to the real observations using the MASE metric. These results are then employed in the decision tree in the next iteration of the classification phase.

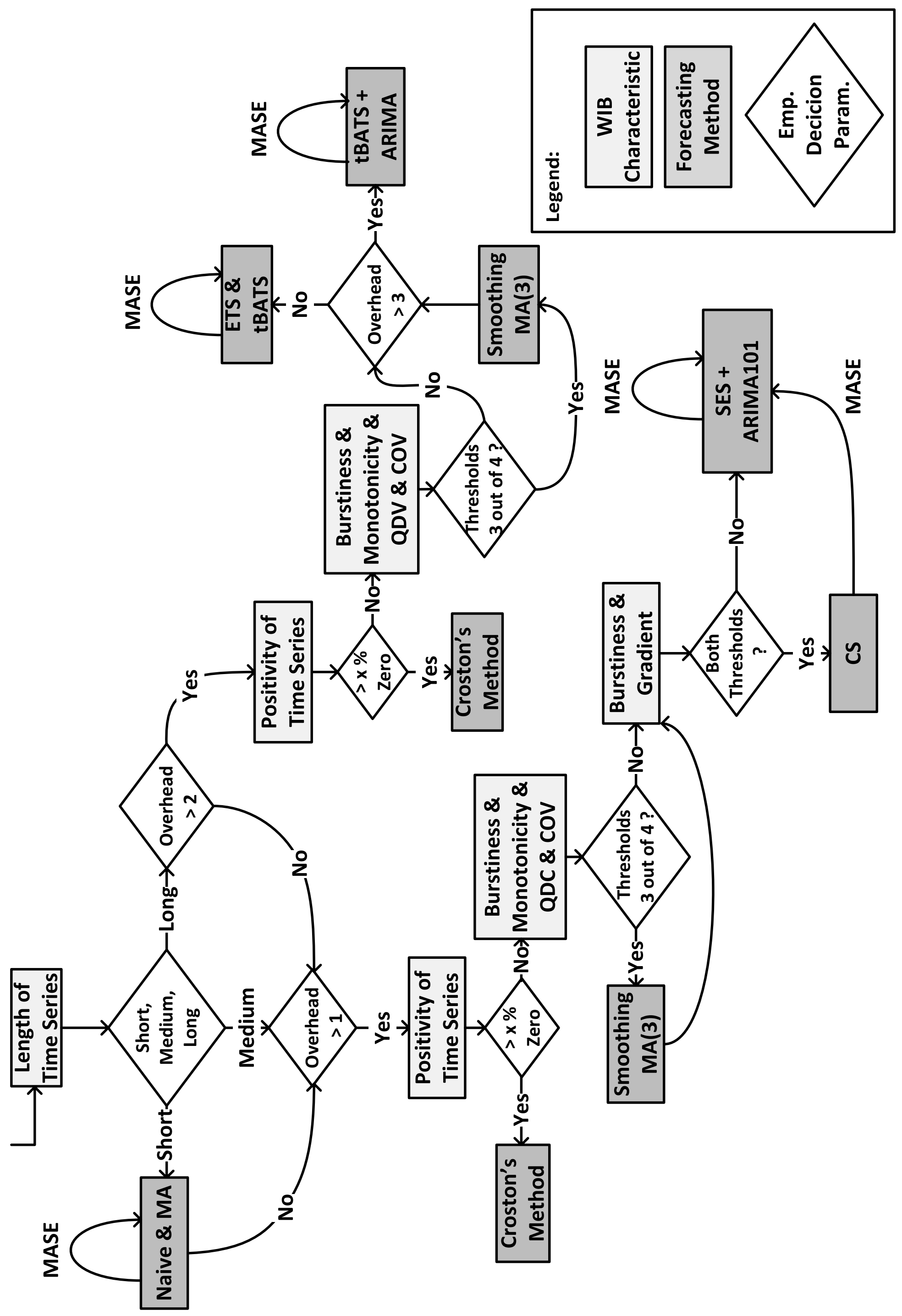
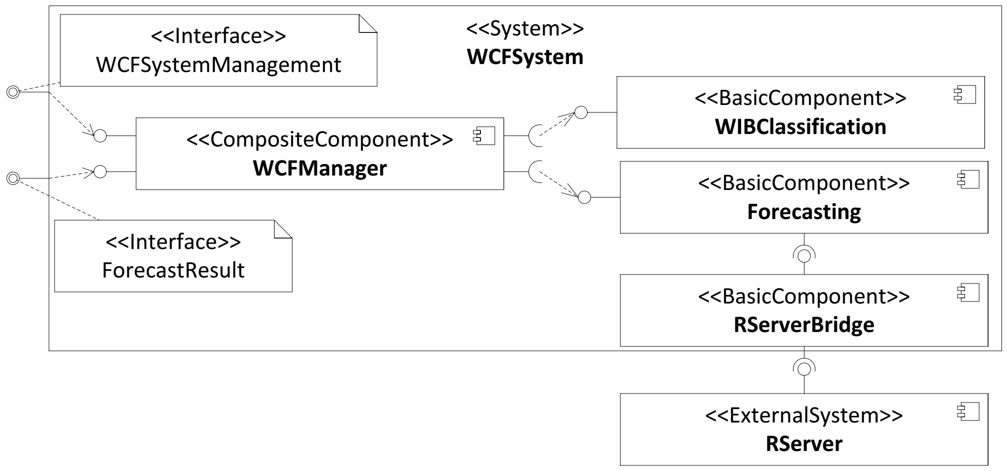


Figure 3. Decision tree for workload classification and selection of appropriate forecasting methods.

4. WCF ARCHITECTURE AND IMPLEMENTATION

We implemented our self-adaptive workload classification and forecasting process in Java. We refer to this implementation as WCF System in the following. For workload forecasting, we used the implementations provided by the forecasting package [¶] for R, a language and environment for statistical computing [25]. Figure 4 depicts the architecture of our WCF System as a UML component diagram.



<<Interface>> WCFSystemManagement
<<System>> WCFSystem
<<BasicComponent>> WIBClassification
<<CompositeComponent>> WCFManager
<<BasicComponent>> Forecasting
<<Interface>> ForecastResult
<<BasicComponent>> RServerBridge
<<ExternalSystem>> RServer

Figure 4. Architecture of the WCF System.

4.1. Architecture

The central component of the WCF System is the `WCFManager`. It is responsible for accepting and managing the *WIB* traces to be classified and forecast and the respective configuration settings. The `WCFManager` periodically triggers the classification and forecasting processes according to the forecasting objectives. In addition, it also implements management functionality to persist the given *WIB* traces, configurations, and results. A more detailed description of how to interact with the `WCFSystem` using its interfaces follows below.

The `WIBClassification` component realizes the classification phase of the WCF process and implements the previously presented decision tree. This component receives a *WIB* trace from the `WCFManager` and selects a forecasting method for this *WIB* according to the decision tree. It returns the result, i.e., the selected WIB class, to the `WCFManager` component.

The `Forecasting` component implements the forecasting phase of our WCF process. It executes the forecasting method that has been selected by the `WIBClassification` component. To obtain forecasting results, the `Forecasting` component uses an external system, the `RServer`. This is basically a TCP/IP server which allows other programs to use the facilities of R, a language and environment for statistical computing [25]. To communicate with the `RServer`, the `Forecasting` component uses the `RServerBride`. This is a wrapper which encapsulates the controlling of R and that processes the results from the R environment.

4.2. Interfaces

The `WCFSystem` provides two interfaces. The `WCFSystemManagement` interface offers management functionality to register new or remove existing *WIBs*, and to read and update the `ForecastingObjectives` of a *WIB*. By setting the forecasting objectives (cf. 3.1), the user can

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

influence the execution and thus the results of the classification and forecast processes. In addition, the `WCFSystemManagement` interface can be used to trigger executions of a classification or a forecast manually.

The second interface provided by `WCFSystem` is the `ForecastResult` interface, which is used for data exchange. By default, data is read from and written to buffers, to support the integration of the `WCFSystem` into a pipes-and-filters architecture. The reason for this design decision is that our WCF system should be able to process data provided at tun-time as an input stream by monitoring frameworks. In this case, input data are newly monitored arrival rate values of the registered *WIBs*, which are provided constantly and in fixed but configurable periods. The output data of the `WCFSystem` are forecast results, which are written into a buffer, too. This buffer can then be used by a resource provisioning system or system adaptation component for further processing. Another option supported by the `WCFSystem` is to write the forecast results to a file for manual result interpretation. More details on the integration of the `WCFSystem` in a model-based adaptation process and system monitoring framework follows below.

4.3. Control Flow

The UML sequence diagram in Figure 5 illustrates the control flow of an exemplary use case of the `WCFSystem`. Note that we omitted the `RServerBridge` in this diagram as it simply acts as a wrapper for the interface of the `RServer`.

To register a new *WIB*, a user can invoke the `registerWIB(ts, forecasting_objectives)` method of the `WCFSystemManagement` interface. This triggers the `WCFManager` to create a new record for this *WIB*. The parameters of this method are a reference to the time series buffer as well as the forecasting objectives for this time series.The referred time series buffer in this context is the entity that continuously provides new monitoring data for the time series.

While the `WCFManager` is active, it periodically triggers the classification and forecasting of its registered *WIBs*, depending on the specified time periods. For classification, the `WCFManager` invokes the `classify` method of the `WIBClassification` component. Arguments are a reference to the time series (`ts`) that shall be classified and the given `forecasting_objectives`. The `WIBClassification` then classifies the given *WIB* according to the process we specified in Section 3 and returns the result `wib_class` to the `WCFManager` component.

To obtain workload forecasts, the `WCFManager` calls the `createForecaster(wib_class)` method of the `Forecasting` component. This creates a `Forecaster` object which acts as a proxy for the forecasting method specified by the `wib_class` argument and implemented in the `RServer`. Then, the `WCFManager` can pass the time series `ts` and the `forecasting_objectives` to the created `Forecaster` to trigger the forecasting process using the `forecast` method. The `Forecaster` then uses the `RServerBride` to communicate with the `RServer` and start the forecast. When finished, it receives the results from the `RServer`, processes and passes them to the `WCFManager`. The `WCFManager` stores the results in the configured location and notifies the user that new forecasting results are available. To remove a *WIB* from the classification and forecasting process, the user can invoke `removeWIB(ts)`. Then, the `WCFManager` component will trigger no further classifications or forecasting.

4.4. Integration

An important requirement for the architecture of the WCF system is that it can be integrated into a pipes-and-filters architecture. The reason is that data must be processed at system run-time. According to [26, p. 53], "the pipes-and-filters architectural pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data is passed through pipes between adjacent filters."

Similarly, in the context of model-based system adaptation (realizing a *MAPE-K* control loop [29]), the MONITOR phase continuously provides new workload data that must be processed at run-time by our `WCFSystem` and directly provided to the ANALYZE phase. Figure 6 sketches a possible integration of the `WCFSystem` into a model-based adaptation control loop [28]. In this

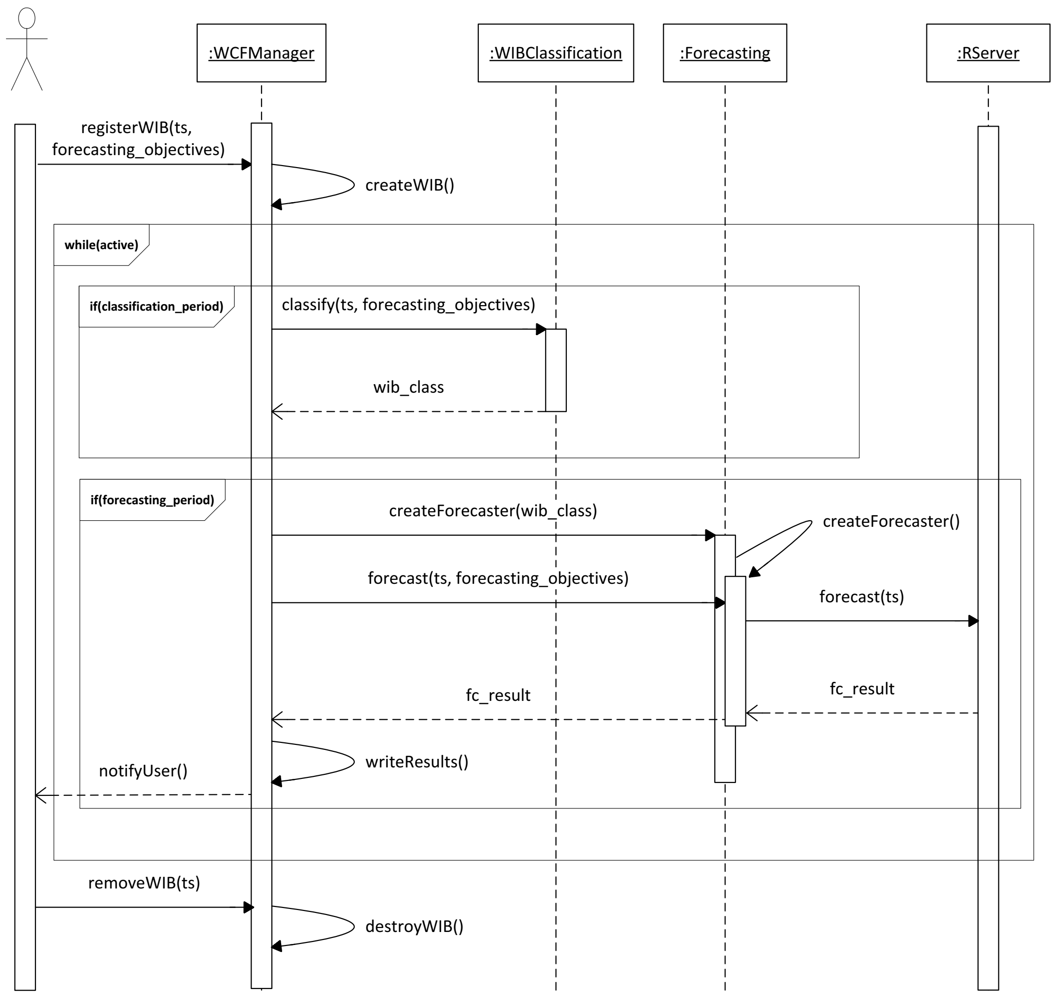


Figure 5. UML sequence diagram illustrating an exemplary use case of the WCF system.

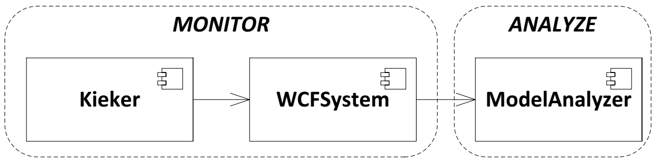


Figure 6. Integration of the `WCFSystem` with an exemplary monitoring framework [27] and a model-based adaptation approach [28].

example, Kieker, a framework for monitoring and analyzing a software system's run-time behavior [27], continuously delivers monitoring data of the service's request arrival rates. These data are piped into our `WCFSystem` that processes the data at run-time and delivers its forecasts to the `ModelAnalyzer` that can use the forecasts to detect problems for the workload forecasts.

5. EVALUATION

In this section, we present results of an extensive evaluation of our WCF approach based on real-world scenarios. Due to the space constraints, we focus on presenting a selected experiment focusing on a forecast accuracy comparison and an exemplary case study where the forecast results are interpreted for proactive resource provisioning. In general, real-world *WIB* traces are likely to exhibit strong seasonal patterns with daily periods due to the fact that the users of many software services are humans. The daily seasonal patterns are possibly overlaid by patterns of a longer periods such as weeks or months. Deterministic bursts within a *WIB* trace are often induced by planned batch tasks (for example in transaction processing systems). In addition, a *WIB* trace can exhibit non-deterministic bursts that cannot be foreseen by any time series analysis technique due to system extrinsic influences. For the evaluation of the WCF approach we used multiple real-world *WIB* traces with different characteristics representing different types of workloads. The results presented in this paper are based on the following two *WIB* traces:

(i) *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[‖] for October 2011.

(ii) *CICS* transaction processing monitoring data was provided by IBM of an real-world deployment of an IBM z10 mainframe server. The data reports the number of started transactions during one week from Monday to Sunday in 30 minute windows.

5.1. Exemplary Scenario

In the presented exemplary scenario, the WCF system uses forecasting methods from all four overhead groups and is compared to the fixed use of the Extended Exponential Smoothing (ETS) method. Additionally, the Naive forecasting method (which is equivalent to system monitoring without forecasting) is compared to the other forecasting methods to quantify and illustrate the benefit of applying forecasting methods against just monitoring the request arrival rates. The individual forecasting methods have been executed with identical exemplary forecast objectives on the same input data and are therefore reproducible. The forecast results are provided continuously over the experiment duration, which means that at the end of an experiment there is for every observed request arrival rate a corresponding forecast mean value with a confidence interval. This allows to evaluate whether the WCF system successfully classifies the *WIB*. A successful classification would mean that the forecasting method that delivers the highest accuracy for a particular forecasting interval is selected by the WCF system. To quantify the forecast result accuracy, a relative error is calculated for every individual forecast mean value:

$$relativeError_t = \frac{|forecastValue_t - observedArrivalRate_t|}{observedArrivalRate_t}.$$

The distributions of these relative errors are illustrated using cumulative histograms which have inclusive error classes on the x-axis and the corresponding percentage of all forecast points on the y-axis. In other words, an $[x, y]$ tuple expresses that y percent of the forecast points have a relative error between 0% and $x\%$. Accordingly, given that the constant line $y = 100\%$ represents the hypothetical optimal error distribution, in each case the topmost line represents the best case of the compared alternatives with the lowest error. We chose to illustrate the error distribution using cumulative histograms to obtain monotone discrete functions of the error distribution resulting in less intersections and therefore in a clearer illustration of the data. To improve readability, the histograms are not drawn using bars but simple lines connecting the individual $[x, y]$ tuples. In addition, statistical key indices like the arithmetic mean, the median and the quartiles as well as the maximum were computed to enable direct comparison of the relative error distributions and rank the forecasting methods according to their achieved forecast accuracy. Finally, directed paired t-tests from common statistics were applied to ensure the statistical significance of the observed differences in the forecast accuracy of the different forecasting methods.

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

This experiment compares the forecast accuracy of the WCF approach with the ETS method and the Naive method during a learning process which means that at the beginning there is no historical data available to the individual forecasting methods that are compared. Concerning the forecasting objectives, the forecast horizon (in this scenario identical to the forecasting frequency) for all forecasting methods is configured to increase stepwise as shown in Table V. The maximum overhead group is set to 4 for an unrestricted set of forecasting methods. The confidence level is set to 95%, but not further interpreted in this scenario.

Table V. Exemplary Scenario

Experiment Focus	Comparison of unrestricted WCF to static ETS and Naive
Forecasting Methods (overhead group)	WCF(1-4), ETS(3), Naive(1)
Input Data WIB trace	CICS transactions, Monday to Friday, 240 values in transactions per 30 minutes, frequency = 48, 5 periods as days
Forecast Horizon (h) (= Forecasting Frequency)	h = 1 for 1^{st} half period, h = 2 until 3^{rd} period complete, h = 12 for 4^{th} and 5^{th} period

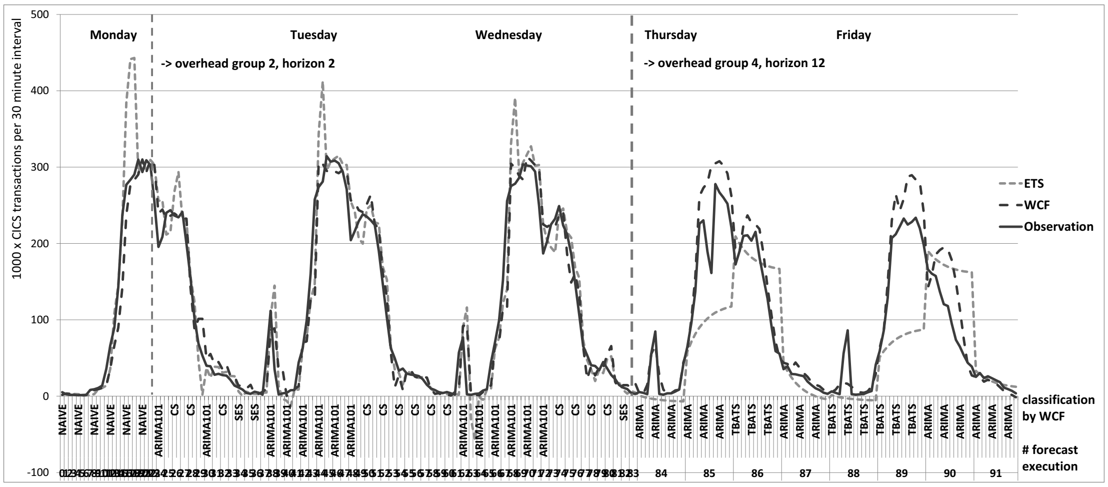
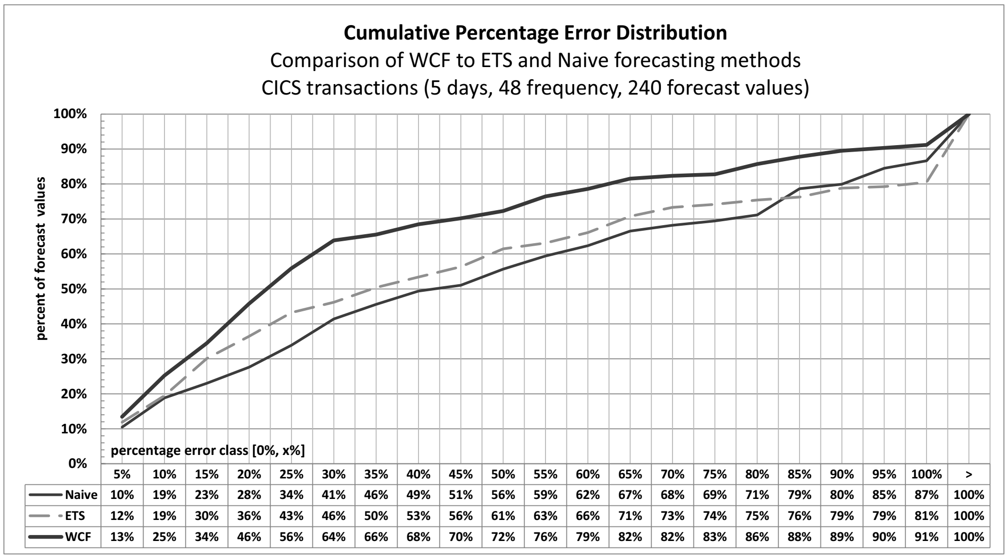


Figure 7. Exemplary Scenario: Comparison Chart of WCF and ETS

The cumulative error distribution for each of the methods are shown in Figure 8 as well as in Figure 9 which demonstrate that the WCF method achieves significantly better forecast accuracy compared to ETS and Naive. ETS can only partially achieve slightly better forecast accuracy than the Naive method, however, it induces processing overheads of 715 ms per forecasting method execution compared to 45 ms for the Naive method (computation of the confidence intervals). The WCF method has an average processing overhead of 61 ms during the first three periods of the *WIB* trace. In the fourth and fifth period (Thursday and Friday), when forecasting methods of the overhead group 4 are selected, WCF's processing overhead per forecasting method execution is on average 13.1 seconds. The forecast mean values of the individual methods and the respective actual observed request arrival rates in the course of time are plotted in Figure 7. In this chart it is visible that the ETS method forecast values have several bursts during the first three periods and therefore do not remain as close to the actual observed values as the WCF forecast values do. During the



	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 8. Exemplary Results: Cumulative Error Distribution of WCF, ETS and Naive

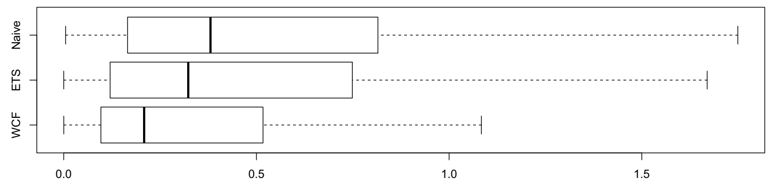


Figure 9. Exemplary Results: Box & Whisker Plots of the Error Distributions

last eight forecast executions the WCF approach successfully detects the daily pattern early enough to achieve better results than ETS. As in this exemplary scenario, we did not observe in any of our experiments with real-world dynamic *WIB*s that the WCF converges to statically select a single forecasting method .

The WCF approach shows the lowest median error value of 20.7% and the lowest mean error value of 47.4%. In addition, WCF has a significantly smaller maximum error value. Though the ETS method is a sophisticated procedure on its own, the results of the presented exemplary scenario demonstrate that the application of ETS on its own leads to poor forecast accuracy with average errors almost as high as for the Naive method and even higher maximal errors.

5.2. *Case Study*

In this section, we present a case study demonstrating how the proposed WCF method can be successfully used for proactive online resource provisioning. WCF offers a higher degree of flexibility due to the spectrum of integrated forecast methods. None of these forecasting methods can offer this degrees of flexibility on their own. For this case study scenario, there is no historic knowledge available at the beginning and the WCF has to adapt to this and wait for the first three periods (Monday, Tuesday, Wednesday) until forecasting methods of the overhead group 4 can be applied. For an arrival rate higher than a given threshold, the SLO of the average response time is violated. We assume that the underlying system of the analyzed workload intensity behavior is linearly scalable from zero to three server instances. This assumption implies a constant average resource demand per request for all system configurations. Furthermore we assume the presence of a resource provisioning system that reacts on observed SLO violations as well as on forecast results. For an arrival rate lower than a given threshold, the running server instances are not efficiently used and therefore one of them is shut down or put in stand by mode. In this scenario, the WCF system provides forecast mean values and confidence intervals to the resource provisioning 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 SLO violations. Details on the WCF configuration for this case study scenario are given in Table VI.

Table VI. Case Study Scenario

Forecast Strategy (overhead group)	WCF(1-4)
Input Data WIB trace	Wikipedia Germany, 3 weeks, 504 values in page requests per hour, frequency = 24, 21 periods as days

Forecast Horizon
(= Forecasting Frequency)
(number of forecast points (h))
	h = 1 for 1^{st} half period
	h = 3 until 3^{rd} period is complete
	h = 12 from 4^{th} on

Confidence Level	80 %

In Figure 10, the WCF forecast values are plotted together with the corresponding confidence intervals and the actual observed workload intensities. The two dotted lines represent the given thresholds that define when a server instance needs to be started or shut down. 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).

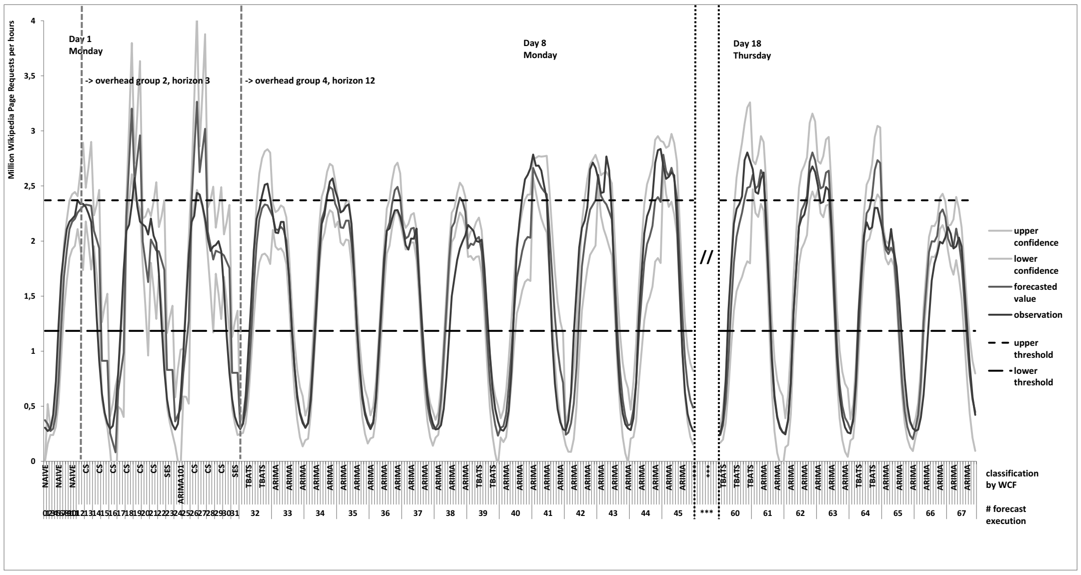


Figure 10. Case Study Chart: 21 Days Wikipedia Page Requests, WCF4 approach

The additions or removals of server instances are triggered based on simple threshold-based rules either in a reactive manner after an SLO violation or in a proactive fashion anticipating the points in time when SLO violations would occur. It can be seen that SLO violations cannot be forecast reliably in the first three daily periods. For the following daily periods, SLO violations can be correctly anticipated in the majority of cases. Only when the amplitude of the daily pattern changes, for example before and after weekends, the forecast mean values deliver false positives or do not anticipate the need for additional computing resources on time. To obtain the results summarized in

Table VII. Case Study: SLO Violations Summary

Comparison	Reactive vs. proactive WCF based provisioning of server instances
Reactive	76 SLO violations
Proactive (WCF based)	42 SLO violations correctly anticipated and 15 more nearly correctly 6 cases of false positives 13 cases of false negatives (not detected)

Table VII, we counted correct, incorrect and nearly correct forecasts for all threshold crossings of the real workload. In the worst case 34 and in the best case 19 SLO violations occur when using the proactive approach based on WCF compared to 76 SLO violations in the reactive case. 17% of SLO violations were not anticipated by the proactive WCF-based approach in addition to 8% reported false positives. The total uptime of the server instances remains equal to a solely reactive provisioning. The start and stop of the server instances is timed more accurately with the result of avoiding the majority of SLO violations and increasing efficiency in resource usage.

5.3. Result Interpretation

As part of our comprehensive evaluation, we considered a number of different scenarios under different workloads with different forecasting objectives and different respective WCF configurations. The experiment results show that the WCF system is able to sensitively select appropriate forecasting methods for particular contexts thereby improving the overall forecast accuracy significantly and reducing the number of outliers as compared to individual forecasting methods applied on their own. In the presented exemplary scenario, WCF achieved an improvement in forecast accuracy on average by 37%. As demonstrated in the case study scenario, the interpretation of WCF forecast results by proactive resource provisioning reduces the number of SLO violations by between 55% to 75%. In addition, our proposed self-adaptive WCF approach supports the system users to select a forecasting method according to their forecasting objectives. Especially at the beginning of a *WIB*'s lifetime, when no or few historical data is available, a static decision made by a user would not fit for the *WIB*'s lifetime. With its dynamic design and the flexibility to react on changes in the *WIB*, the WCF system is able to adapt to such changes, thereby increasing the overall accuracy of the forecast results. Our WCF system enables online and continuous forecast processing with controllable computational overheads. We achieve this by scheduling forecasting method executions and *WIB* classifications in configurable periods. In all experiments, the processing times of all forecasting methods remained within the boundaries specified by their corresponding overhead group in such a way that the forecast results are available before their corresponding request arrival rates are monitored. The latter is crucial especially for the execution of forecasting methods at a high frequency with short horizons.

For a *WIB* with strong daily seasonal pattern and high amplitude variance due to known calendar effects, the forecast accuracy might be strongly improved by splitting this *WIB* into two separate time series: regular working days in the first and weekends and public holidays in the second. This can reduce the impact of the possibly strong overlay of weekly patterns. As part of our future work, we plan to introduce support for such time series splitting as well as for the selection of the data aggregation level for varying forecasting objectives. This can be achieved by a realisation of an intelligent filter applied to the monitoring data before it is provided to the WCF system in form of time series.

6. RELATED WORK

We divide related work into the following three groups: (i) The first group of related work has its focus on evaluating forecasting methods applied to either workload related data or performance monitoring data. (ii) In the second group of related work we consider approaches for workload forecasting that are not based on the methods of the time series analysis, as opposed to our proposed WCF approach. (iii) In the third group of related work we cover research that has its focus on approaches for proactive resource provisioning using tailored forecasting methods.

(i) In 2004, Bennani and Menasce have published their research results of an robustness assessment on self-managing computer systems under highly variable workloads [30]. They come to the conclusion that proactive resource management improves a system's robustness under highly variable workloads. Furthermore, the authors compare three different trend interpolating forecasting methods (polynomial interpolation, weighted moving averages and exponential smoothing) that have been introduced as means for workload forecasting in [31]. Bennani and Menasce propose to select the forecasting methods according to the lowest R^2 error as accuracy feedback. Still, the focus of this work is more on the potentials of proactive resource provisioning and the evaluation of forecasting methods is limited to basic trend interpolation methods without any pattern recognition for seasonal time series components. In [32], Frotscher assesses the capabilities to predict response times by evaluating the forecast results provided by two concrete ARIMA models without seasonality, simple exponential smoothing (SES) and the Holt-Winters approach as a special case of the extended exponential smoothing (ETS). His evaluation is based on generated and therefore possibly unrealistic times series data. The author admits that the spectrum of forecasting methods offered by time series analysis is not covered and he 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.

(ii) In [33] and a related research paper [34], the authors propose 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 [35]. Shortcoming of these approaches is that 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 high amounts of monitoring data for the mining of possibly observable patterns.

(iii) The authors of [3] use a tailored method to decompose a time series into its dominating frequencies using a Fourier transformation and then apply trend interpolation techniques to finally generate a synthetic workload as forecast. Similarly in [4], Fourier analysis techniques are applied to predict future arrival rates. In [8], 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 [5] 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 [7], 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 [36, 37] 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 frameworks 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. In [6], different methods of the time series analysis and Bayesian learning approaches are applied. The authors propose to periodically selected a forecasting method using forecast accuracy metrics. But it is not further evaluated how significantly this feedback improves the overall forecast accuracy. No information is given on how the user's forecast objectives are captured and on how the computational overheads can be controlled. In contrast to our research, the authors concentrate on the prediction of resource consumptions.

However, besides of [6], a common limitation of the above mentioned approaches is their focus on individual methods that are optimised or designed to cover a subset of typical situations. Therefore, they are not able to cover all possible situations adequately as it could be achieved by an intelligent combination of the innovation state space frameworks (ETS and tBATS) and the auto-regressive integrated moving averages (ARIMA) framework for stochastic process modeling. Furthermore, in all mentioned approaches besides in [4], the resource utilisation or average response times are monitored and taken as an indirect metric for the recent workload intensity. But these QoS metrics are indirectly influenced by the changing amounts of provisioned resources and several other factors in a dynamic environment. The forecasting computations are then based on these values that bypass resource demand estimations per request and interleave by principle performance relevant characteristics of the software system as they can be captured in a system performance model.

7. CONCLUSION

Today's resource managing systems of virtualized computing environments often work solely reactive using threshold-based rules, not leveraging the potential of proactive resource provisioning to improve a system performance while maintaining resource efficiency. The results of our research is a step towards exploiting this potential by computing continuous and reliable forecasts of request arrival rates with appropriate accuracy.

The basis of our approach was the identification of *Workload Intensity Behavior* specific characteristics and metrics that quantify them. Moreover, we presented a survey on the strengths, weaknesses and requirements of existing forecast methods from the time series analysis and ways to estimate and evaluate the forecast accuracy of individual forecast executions. As the major contribution of this paper, we presented an approach classifying *Workload Intensity Behaviors* to dynamically select appropriate forecasting methods. This has been achieved by using direct feedback mechanisms that evaluate and compare the recent accuracy of different forecasting methods. They have been incorporated into a decision tree that considers user specified forecasting objectives. This enables online application and processing of continuous forecast results for a variable number of different *WIB*s. Finally, we evaluated an implementation of our proposed approach in different experiments and an extensive case study based on different real-world workload traces. Our experiments demonstrate that the relative error of the forecast points in relation to the monitored request arrival rates is significantly reduced as in the exemplary scenario by 37% on average compared to the results of a static application of the Extended Exponential Smoothing (ETS) method. More importantly, the major benefit of our approach has been demonstrated in an extensive case study, showing how it is possible to prevent between 55% and 75% of SLO violations.

7.1. Future Work

We plan to extend the functionality of the WCF system to apply it in further areas. One 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 forecasting objectives, e.g, continuous short term forecasts for proactive resource allocation or manually triggered long term forecasts for server capacity planning. For divergent forecasting objectives, such a filter would multiplex an input stream of monitoring data and provide it as individual *WIB*s in different aggregation levels at the data input interface of the WCF system. A second WCF system application area will be a combination with an anomaly detection system like ΘPAD as outlined in [38]. Such a system can compute continuous anomaly ratings by comparing the WCF system's forecast results with the monitored data. The anomaly rating can serve to analyze the *Workload Intensity Behavior* for sudden and unforeseen changes and in addition as an reliability indicator of the WCF system's recent forecast results.

Ultimately, we will connect the WCF system's interface (data input) to a monitoring framework like Kieker [27] to continuously provide online forecast results. These forecast results can then be used as input for self-adaptive resource management approaches like in [28], giving them the ability to proactively adapt the system to changes in the workload intensity. This combination of workload forecasting and model-based system reconfiguration is an important step towards the realization of self-aware systems [39].

REFERENCES

1. Abdelsalam H, Maly K, Mukkamala R, Zubair M, Kaminsky D. Analysis of energy efficiency in clouds. *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World:*, 2009; 416–421, doi:10.1109/ComputationWorld.2009.38.
2. Lefvre L, Orgerie AC. Designing and evaluating an energy efficient cloud. *The Journal of Supercomputing* 2010; **51**:352–373. URL http://dx.doi.org/10.1007/s11227-010-0414-2, 10.1007/s11227-010-0414-2.
3. Gmach D, Rolia J, Cherkasova L, Kemper A. Workload analysis and demand prediction of enterprise data center applications. *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, IISWC '07, IEEE Computer Society: Washington, DC, USA, 2007; 171–180, doi:http://dx.doi.org/10.1109/IISWC.2007.4362193. URL http://dx.doi.org/10.1109/IISWC.2007.4362193.
4. Hedwig M, Malkowski S, Bodenstein C, Neumann D. Towards autonomic cost-aware allocation of cloud resources. *Proceedings of the International Conference on Information Systems ICIS 2010*, 2010. URL http://aisel.aisnet.org/icis2010_submissions/180/.
5. Bobroff N, Kochut A, Beaty K. Dynamic placement of virtual machines for managing sla violations. *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 2007; 119 –128, doi:10.1109/INM.2007.374776.
6. Meng X, Isci C, Kephart J, Zhang L, Bouillet E, Pendarakis D. Efficient resource provisioning in compute clouds via vm multiplexing. *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, ACM: New York, NY, USA, 2010; 11–20, doi:http://doi.acm.org/10.1145/1809049.1809052. URL http://doi.acm.org/10.1145/1809049.1809052.
7. Kim H, Kim W, Kim Y. Predictable cloud provisioning using analysis of user resource usage patterns in virtualized environment. *Grid and Distributed Computing, Control and Automation*, *Communications in Computer and Information Science*, vol. 121, Kim Th, Yau SS, Gervasi O, Kang BH, Stoica A, Slezak D (eds.). Springer Berlin Heidelberg, 2010; 84–94.
8. Caron E, Desprez F, Muresan A. Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. *J. Grid Comput.* March 2011; **9**:49–64, doi:http://dx.doi.org/10.1007/s10723-010-9178-4. URL http://dx.doi.org/10.1007/s10723-010-9178-4.
9. Temple J, Lebsack R. Fit for Purpose: Workload Based Platform Selection. *Technical Report*, IBM Corporation 2010.
10. Mitsa T. *Temporal Data Mining*. 1st edn., Chapman & Hall/CRC, 2010.
11. Calzarossa M, Serazzi G. A characterization of the variation in time of workload arrival patterns. *IEEE Trans. Comput.* February 1985; **34**:156–162, doi:10.1109/TC.1985.1676552. URL http://dl.acm.org/citation.cfm?id=1309285.1309718.
12. Arlitt MF, Williamson CL. Web server workload characterization: the search for invariants. *SIGMETRICS Perform. Eval. Rev.* May 1996; **24**:126–137.
13. Schroeder B, Wierman A, Harchol-Balter M. Open versus closed: a cautionary tale. *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, USENIX Association: Berkeley, CA, USA, 2006; 18–18. URL http://dl.acm.org/citation.cfm?id=1267680.1267698.
14. van Hoorn A, Rohr M, Hasselbring W. Generating probabilistic and intensity-varying workload for web-based software systems. *Proceedings of the SPEC international workshop on Performance Evaluation: Metrics, Models and Benchmarks*, SIPEW '08, Springer-Verlag: Berlin, Heidelberg, 2008; 124–143.
15. Ali-Eldin A, Tordsson J, Elmroth E, Kihl M. Workload classification for efficient auto-scaling of cloud resources 2013.
16. Box G, Jenkins G, Reinsel G. *Time series analysis : forecasting and control*. 4. ed. edn., Wiley series in probability and statistics, Wiley: Hoboken, NJ, 2008. Includes index.
17. Hyndman R, Khler A, Ord K, Snyder R ((eds.)). *Forecasting with Exponential Smoothing : The State Space Approach*. Springer Series in Statistics, Springer-Verlag Berlin Heidelberg: Berlin, Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-71918-2.
18. Shumway RH. *Time Series Analysis and Its Applications : With R Examples*. Springer Texts in StatisticsSpringerLink : Bcher, Springer Science+Business Media, LLC: New York, NY, 2011. URL http://dx.doi.org/10.1007/978-1-4419-7865-3.
19. Verbesselt J, Hyndman R, Zeileis A, Culvenor D. Phenological change detection while accounting for abrupt and gradual trends in satellite image time series. *Remote Sensing of Environment* 2010; **114**(12):2970 – 2980, doi:10.1016/j.rse.2010.08.003. URL http://www.sciencedirect.com/science/article/pii/S0034425710002336.
20. Hyndman RJ, Khandakar Y. Automatic time series forecasting: The forecast package for R 7 2008. URL http://www.jstatsoft.org/v27/i03.
21. De Livera AM, Hyndman RJ, Snyder RD. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association* 2011; **106**(496):1513–1527, doi:10.1198/jasa.2011.tm09771. URL http://pubs.amstat.org/doi/abs/10.1198/jasa.2011.tm09771.
22. Hyndman RJ, King ML, Pitrun I, Billah B. Local linear forecasts using cubic smoothing splines. *Monash Econometrics and Business Statistics Working Papers 10/02*, Monash University, Department of Econometrics and Business Statistics 2002. URL http://EconPapers.repec.org/RePEc:msh:ebswps:2002-10.

23. Shenstone L, Hyndman RJ. Stochastic models underlying croston's method for intermittent demand forecasting. *Journal of Forecasting* 2005; **24**(6):389–402, doi:10.1002/for.963. URL http://dx.doi.org/10.1002/for.963.
24. Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. *International Journal of Forecasting* 2006; :679–688.
25. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria 2011. URL http://www.R-project.org/, ISBN 3-900051-07-0.
26. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley: Chichester, UK, 1996.
27. van Hoorn A, Waller J, Hasselbring W. Kieker: A framework for application performance monitoring and dynamic software analysis. *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, ACM, 2012; 247–248.
28. Huber N, van Hoorn A, Koziolek A, Brosig F, Kounev S. Modeling Run-Time Adaptation at the System Architecture Level in Dynamic Service-Oriented Environments. *Service Oriented Computing and Applications Journal (SOCA)* 2013; doi:10.1007/s11761-013-0144-4.
29. Kephart JO, Chess DM. The vision of autonomic computing. *Computer* 2003; **36**(1):41–50.
30. Bennani MN, Menasce DA. Assessing the robustness of self-managing computer systems under highly variable workloads. *Proceedings of the First International Conference on Autonomic Computing*, IEEE Computer Society: Washington, DC, USA, 2004; 62–69. URL http://dl.acm.org/citation.cfm?id=1078026.1078409.
31. Menascé DA, Almeida VAF. *Capacity planning for Web performance: metrics, models, and methods*. Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1998.
32. Frotscher T. Prognoseverfahren für das Antwortzeitverhalten von Software-Komponenten March 2011. Christian-Albrechts-Universität zu Kiel, Bachelor's Thesis.
33. Kleeberg SD. Neuronale Netze und Maschinelles Lernen zur Lastvorhersage in z/OS. Universität Tübingen, Diploma Thesis.
34. Bensch M, Brugger D, Rosenstiel W, Bogdan M, Spruth WG, Baeuerle P. Self-learning prediction system for optimisation of workload management in a mainframe operating system. *ICEIS*, 2007; 212–218.
35. Goldszmidt M, Cohen I, Powers R. Short term performance forecasting in enterprise systems. *In ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, ACM Press, 2005; 801–807.
36. Amin A, Grunske L, Colman A. An Automated Approach to Forecasting QoS Attributes Based on Linear and Non-linear Time Series Modeling. *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE/ACM, 2012.
37. Amin A, Colman A, Grunske L. An Approach to Forecasting QoS Attributes of Web Services Based on ARIMA and GARCH Models. *Proceedings of the 19th IEEE International Conference on Web Services (ICWS)*, IEEE, 2012.
38. Bielefeld TC. Online Performance Anomaly Detection for Large-Scale Software Systems March 2012. Diploma Thesis, University of Kiel.
39. Kounev S, Brosig F, Huber N, Reussner R. Towards self-aware performance and resource management in modern service-oriented systems. *Proceedings of the 2010 IEEE International Conference on Services Computing*, SCC '10, IEEE Computer Society: Washington, DC, USA, 2010; 621–624, doi:http://dx.doi.org/10.1109/SCC.2010.94. URL http://dx.doi.org/10.1109/SCC.2010.94.