Architecture-Based Multivariate Anomaly Detection for Software Systems

Master's Thesis

Tom Frotscher

October 16, 2013

Kiel University
Department of Computer Science
Software Engineering Group

Advised by: Prof. Dr. Wilhelm Hasselbring
Dipl.-Inf. Jan Waller
Dipl.-Inform. André van Hoorn (Univ. of Stuttgart)
Dr. Stefan Kaes (Xing AG)

Abstract

With growing software systems appropriate monitoring concepts become more and more important. For large-scale software systems this becomes a problem, simply because of the overwhelming size and complexity these systems can reach. Performance data is of special interest in this context, since it is related to factors such as the confidence and satisfaction customers have in the offered services.

In many cases, performance issues manifest themselves in so-called anomalies, that do not fit to the normal behavior of the software system. ΘPADx, an approach and a corresponding implementation, that is developed in the process of this thesis, is able to recognize such anomalies. In contrast to often used offline analysis approaches, the ΘPADx approach is able to process performance data online, which makes fast reactions to the anomalies and the correlated issues possible.

ΘPADx is based on an approach called ΘPAD, that was presented by Bielefeld in 2012. An analysis of ΘPAD reveals weaknesses of the approach, whenever anomalies are occurring over a long time period. Based on this analysis, extensions are developed and implemented to improve the approach. With the help of the implementation, the extended approach, then named ΘPADx, is evaluated on past and actual data of the social network Xing. Xing serves as case study environment for the actual thesis and also for the work of Bielefeld, which makes the results comparable. It turns out, that the extensions lead to improved anomaly detection results and an enhanced usability of the approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the past ten years the Internet and its possibilities developed rapidly. Between 2010 and today, the user count grew by more than 550%, as can be seen at the Internet World Stats Website [Miniwatts 2013]. Since the Web 2.0 is arising, services like social networks, blogs or photo-sharing-networks became popular. As an example, Facebook stands out with an increase from one million users in 2004 to more than one billion users in 2012 [Facebook 2013]. Backed up by an increasing amount of fast Internet connections and mobile devices, a huge number of users are "all time" connected to the Internet. Therefore, services that are available via Internet and are hosted on distant data centers became the solution of choice for many companies [Limam 2010]. Such services are summarized under the term *Software as a Service* (SaaS). There are several advantages of the SaaS approach. For example, the software is easily distributed via Internet and maintenance-free at the user side [Limam 2010; Ju 2010]. With the distributed architecture used by SaaS however, some disadvantages arise as well. One key factor for the customer satisfaction is the performance of the used software. The performance is one of three factors, which are summarized under the term *Quality of Service* (QoS). As defined by Becker [Becker et al. 2006] for example, QoS consists of the factors *availability*, *reliability*, and *performance*.

The factors of QoS can be affected by different circumstances. The huge amount of users in the Internet or the fact, that trends or phenomena are propagated quickly through the web due to the popularity of social networking, can lead to huge workloads and huge variations in workloads. The hardware resources behind services are often bounded or can not been allocated in time. The factors of QoS can also be affected by attacks against applications or by hardware problems. Faulty behavior, slow response times, or even the outage of the service might be the consequences. These situations can bring up several problems for the software provider. Customers might switch to competitors' products or the company might get involved in costly law suits, caused by violated *Service-Level-Agreements* (SLAs). Therefore, a continuous observation of the application and its underlying system is important. The possibility of detecting abnormal behavior of a system with the help of these kinds of observations called *monitoring*, is a key issue that is considered in this thesis.

Anomalies are patterns in data that are not conform to the majority of the containing data set. The detection of such patterns is a problem that is addressed in many different research areas. The analysis of network traffic or the detection of credit card fraud are just a few examples [Chandula et al. 2009]. Beside more generic methods, special detection methods were often invented for certain research domains [Chandula et al. 2009]. One possibility to find anomalies relies on the so-called *time series analysis*. The data of an observed variable can be transformed to a time series and processed with algorithms like exponential smoothing or "averaging methods". The result is a new time series that can be compared to the original one.

The *Online Performance Anomaly Detection* (ΘPAD) [Bielefeld 2012a] approach uses time series analysis based methods for its anomaly detection. ΘPAD is a concept and a corresponding implementation introduced by Tillmann Carlos Bielefeld in 2012 [Bielefeld 2012a]. This approach is able to achieve automatic detection of performance anomalies. As a case study, ΘPAD was integrated and evaluated at the social network system *Xing*[1], where it reached results of good quality. However, some disadvantages arose with a special kind of anomaly [Bielefeld, p.94] and the usage of ΘPAD for systems that offer performance data from multiple sources. While the detection of short term anomalies with ΘPAD is reliable, anomalies that occur over a longer time period caused additional alerts whenever the system returns to its normal behavior. Since the approach supposes an abnormal behavior while the system runs normal, these are false alerts. Because of this fact, the practical usage of the ΘPAD approach is disturbed.

This thesis aims to investigate a solution for these disadvantages by extending the anomaly detection algorithm of ΘPAD. Xing is chosen as case study environment to reach a high comparability within the evaluation. Since ΘPAD was developed, some changes to the environment of Xing were made. The main application of the Xing network got split in several smaller, interdependent applications. While ΘPAD was able to run an analysis for a single application, an analysis process must now be able to handle the input of several applications. The approach and the corresponding implementation, which are the result of this thesis, therefore also extend the ΘPAD approach to support the anomaly detection on performance data of multiple sources. With respect to the extensions that will arise throughout this thesis, the resulting approach will be called *Extended Online Performance Anomaly Detection* (ΘPADx).

1.2 Xing - Overview

Xing is a social network system hosted by Xing AG. Since the company was founded in 2003 under the name openBC, the user base of the social network grew fast. In 2006 the

[1] Xing Website: http://www.xing.com

company was renamed to Xing. Nowadays, over 13 million members worldwide and about 6.5 million users in German-speaking countries are registered on Xing. This makes Xing Europe's most important network aiming on business contacts [Xing AG 2009]. Users are able to share their profiles, create groups, or organize events via Xing to search and find expert knowledge, jobs, employees, or ideas [Xing AG 2012]. Additional features are available for members who possess a payed premium membership. With currently over 800,000 premium members, this builds the company's most important source of revenue [Xing AG 2013a].

The member base, steadily growing since 2003, leads to a continuous progression of the platforms software. With the enhanced requirements due to the growing amount of users, the software evolved during the years. In 2003 the software was written in Perl and was hosted on only a few servers. Today, the platform consists of multiple applications written in *Perl* and *Ruby on Rails* (see Section 2.4.3) hosted on hundreds of physical servers in two data centers. To obtain the platform's performance even under the growing amount of processed data, new technologies, as for example *eventual consistency* [Gustavsson and Andler 2002] or *data base sharding* [Chodorow 2011], are established over the time.

While the platform becomes more and more complex, an appropriate performance monitoring solution is necessary. Xing already uses the performance monitoring tool Logjam (see Section 2.4.3), which is able to store and visualize a wide range of performance data. However, it is still a hard task to recognize performance issues in time, due to the huge amount of components and applications the software system contains of. OPADx aims to solve this problem more reliably than OPAD, while using a similar, but improved automatic detection approach.

1.3 Goals

The motivation of this thesis already sketched what kind of extensions are necessary for the ΘPAD approach. Therefore, the goals derived from the motivation section and are presented in a more detailed manner. For a better over all structure the thesis is oriented according to the *Goal Question Metric Paradigm* developed by Basili and Caldiera [Basili et al. 1994]. This paradigm is also briefly introduced within this section.

1.3.1 Goal Question Metric Paradigm

The *Goal Question Metric Paradigm* (GQM) was originally developed to improve processes in the software engineering domain. Most of the time the quality of processes and the corresponding results are measured with metrics that are easy to collect, for example the number of code lines in a software project or the number of pages of a thesis. But these metrics are hard to interpret without a corresponding context. Therefore, the GQM

approach follows a more top-down oriented structure. First, *goals* are defined. Then a set of *questions* is created for every goal to gain a more detailed characterization of it. With the context given by the goals and questions, a fitting *metric* can be derived eventually. Following the Goal Question Metric Paradigm, such a metric is way more meaningful than metrics that are not correlated with the given context. These three parts form a hierarchical structure visualized in Figure 1.1, the so-called *GQM model*.

Goal

In the first step the goals are defined. The GQM model defines two different kinds of goals in general. The goals on the *organisational* level represent the needs and interests of the organization or person that will develop the covered process. These can be for example "we need to put more effort in the customer service" or "we need to improve our overall productivity". For this thesis an example might be "i need to improve the anomaly detection approach".
The second kind of goals are the *conceptual* goals, defined to split up and solve the overall goals. A conceptual goal is set for a specific *object* and addresses an *issue* and a corresponding *purpose* related to the object. The complete goal is defined under a certain *viewpoint*. This means that different stakeholders may have different expectations regarding the goal. An example in the context of the improvement of the anomaly detection approach can then look as followed: "find a good configuration setup of ΘPADx for the evaluation environment". Where the purpose is to find the configuration, the issue is the configuration itself, the object is ΘPADx and the viewpoint is the case study environment.

Question

For a more detailed specification of the way to achieve a result for a defined conceptual goal, so-called *questions* are used within the GQM approach. Every question aims on a certain quality issue of the measured object under a certain viewpoint. With the help of the questions the respective issue can be divided into its main components. In order to be able to answer the questions in a quantitative way, a set of data is associated with every question. Following the example, a question could be "What is the best forecasting method to chose?"

Metric

Each question is refined in different metrics. A metric can either be *objective* or *subjective*. While an objective metric depends only on the object itself, a subjective metric depends on the object and the viewpoint. For example the lines of code are an objective metric, while the readability of the code is a subjective metric. This is because the readability depends also on the person who reads the code. An example metric to assess the quality of a detection approach is the F-measure, that will be introduced in the foundations chapter.

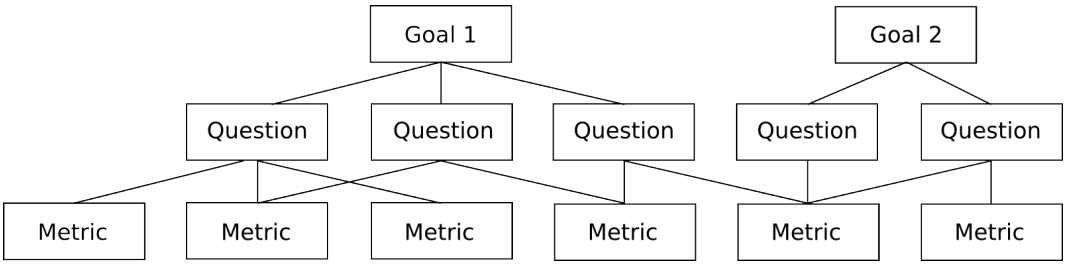

Figure 1.1. Overview of the GQM process. (Image based on [Basili et al. 1994])

1.3.2 Organizational Goals

Within this section the goals are defined as described by the GQM approach. The points mentioned here are the goals for the entire thesis. This will correspond to the goals on the organisational level in the GQM domain. In the evaluation process (Chapter 5) the conceptional goals will take up the goals again and divide them into the main components with the help of evaluation questions. This offers the opportunity to derive fitting metrics. Better interpretable research results are expected with this approach.

- G1: Improvement of the ΘPAD anomaly detection approach

  Develop a way to improve the anomaly detection approach of ΘPAD. To accomplish this, a hybrid method of two kinds of forecasters is used. Furthermore, the parameters of the algorithms and the anomaly detection have to be readjusted to match the new approach. Additionally, the anomaly detection will be designed to work on multiple applications and metrics (see G3.1 and G3.2).

- G2: Implementation of the ΘPADx approach

  The Kieker plugin architecture has been reworked since ΘPAD was developed. An implementation of the extensions to the ΘPAD approach – still based on Kieker and also matching the new architecture – should be created. Furthermore, the implementation should take the additional forecasting method, used for the improvement, into account. Therefore, it must be able to import and process information from a long-term data set and combine it with the *forecasts* to achieve an improved anomaly detection.

- G3: Integration of the extended approach in the redefined Xing environment

  Since the ΘPAD approach has been introduced, the environment at Xing changed. This leads to the following extension of ΘPAD:

- ▻ G3.1: A new transport layer is introduced. Instead of *AMQP* now the brokerless *ZeroMQ* is chosen. To be able to process the data from the messaging queue, the adapter has to be adjusted to support ZeroMQ.

Additionally, two conceptional changes are necessary:

- ▻ G3.2: The application monitored with ΘPAD is divided into several individual and inter-dependent applications. A way to read and separate the data from the incoming messages of multiple applications has to be implemented to adapt to the architectural changes.
- ▻ G3.3: Finally, the applications now give information about a huge number of metrics like *response time*, *cpu load* or *concurrent users*. Therefore, the implementation has to support the multiple *metrics* mentioned in Chapter 2. This leads to the implementation of corresponding new *Monitoring Records* to process these kinds of data within the *Kieker Framework*.

- ▻ G4: Evaluation

The ΘPADx approach can still underlie variations in terms of *accuracy* and *precision* [Salfner et al. 2010]. A long-term test will be made to assess the results of the extensions in respect to accuracy, precision and other metrics (see Section 2.2.1). To ensure the comparability of the approaches, the evaluation is additionally going to involve the same long term data set used in the evaluation of ΘPAD. Finally, it will be determined if the ΘPADx approach is going to produce results of higher quality than ΘPAD.

1.4 Document Structure

After a short introduction and the presentation of the thesis goals in this first chapter, the foundations of the research topic and the used technologies are presented in Chapter 2. In Chapter 3 the approach, chosen to accomplish the formulated goals, is presented. After that, the details of the corresponding implementation are covered in Chapter 4. With the help of the implementation an evaluation takes place, that is described in Chapter 5. In Chapter 6 a comparison with related works is executed. Finally, in Chapter 7 the thesis is summed up and a conclusion is given.

Chapter 2

Foundations

The ΘPADx approach is based on several terms, concepts, and technologies to achieve an automatic anomaly detection, which are introduced within this chapter. The chapter starts with an introduction to time series and the most important forecasting algorithms in Section 2.1. After this a definition of anomalies and an explanation of the chosen approach of anomaly detection is given. Since ΘPADx is an approach focused on performance data, also the performance metrics that are considered are introduced (see Section 2.3). Within this thesis an implementation of the ΘPADx approach is developed as well. The technologies and concepts used for this implementation are therefore presented in Section 2.4. For evaluation purposes, the developed ΘPADx implementation is used within the case study environment at Xing. For this purpose, an introduction to the Xing architecture and the corresponding technologies is given in Section 2.4.3. Finally the ΘPAD approach is presented in Section 2.5, which is required to understand the extensions that are made to the approach and presented in the next chapters.

2.1 Time Series Analysis

Time series data typically arise when a variable is observed over a period of time. The assumption is made, that the points gathered during the observation process may have an internal structure. Therefore, every point is dependent on the current situation and the previous values of the observation. The intention of time series analysis is to discover this internal structure.

2.1.1 Basic Definitions

A monitoring tool mainly observes system variables over a duration of time. *Measurements* of a variable taken at regular temporal intervals lead to the formal definition of *discrete time series* as described by Mitsa [Mitsa 2009]:

A time series $X = \{x_1, x_2, ..., x_n\}$ for $t = t_1, t_2, ..., t_n$ is a discrete function, chronological ordered with value x_1 for time t_1, value x_2 for time t_2, and so on. Whenever the term time series is used within this thesis, a discrete time series is intended. A measurement that is already part of a time series is called *time series point*.

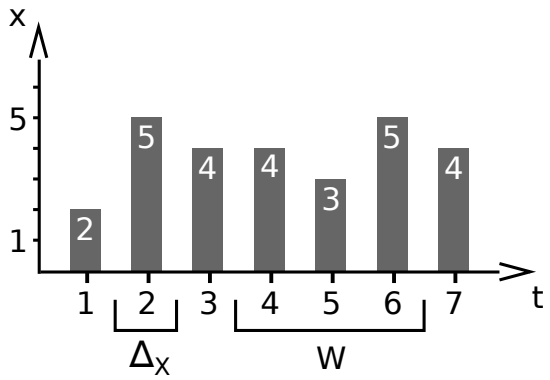

Figure 2.1. Barplot of the time series X with step size Δ_X and a time series window W.

With the given time index t, the *duration* D of a time series can be defined as the distance between the first and the last time series point:

$$D_X = t_n - t_1$$

Furthermore, the distance between two consecutive time series points t_i and t_{i+1} is called *step size*. For discrete time series the step size is always constant and can be calculated by:

$$\Delta_X = t_{i+1} - t_i$$

Finally, a time series can be *univariate* or *multivariate*. While a univariate time series is created by one underlying variable, the amount of variables for multivariate time series is greater than one.

In the context of forecasting methods often only a part of a times series is taken into account. For this thesis such a part of a time series $X = \{x_1, x_2, ..., x_n\}$ is called *time series window* $W = \{w_1, w_2, ..., w_m\}$. Therefore, the following constraints must hold:

- $W \subseteq X$
- the step size of X must equal the step size of W
- the values in the time window W must be equal to the corresponding values in X

$$\exists h \in \mathbb{N}, \forall i \in 1, ..., m : x_{h+i} = w_i$$

A simple way to visualize time series can be achieved by *X-Y-Plots*. An example visualisation of the fictitious time series $X = \{2, 5, 4, 4, 3, 5, 4\}$ with $\Delta_X = 1$ is given in Figure 2.1.

According to [Verbesselt et al. 2010; Box and Jenkins 1990; Shumway 2011] time series data can be decomposed in three different components: *trend*, *season* and *remainder*.

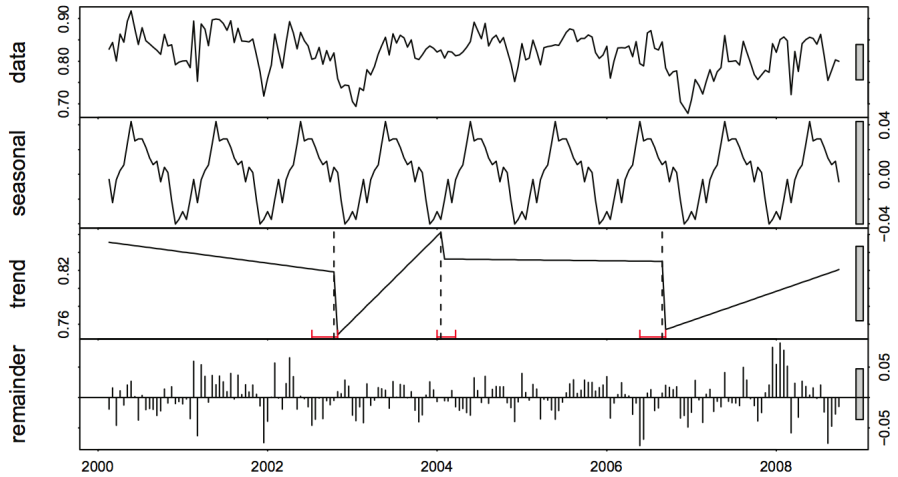

Figure 2.2. The time series (data) is decomposed in the trend, season and remainder components. This decomposition was presented in [Verbesselt et al. 2010].

A model that is composed of these components is called a *component model*. The trend component can be seen as long-term variation and can be described by a monotonically increasing or decreasing function as stated by [Herbst et al. 2013]. The seasonal component includes patterns that influence the time series in regular time intervals and in similar extends. For example the workload of a website is lower at night while most users are asleep. This pattern is reoccurring every day. Similar patterns can occur for different time units, for example weekly or monthly. Finally, the remainder component is interpreted as a random value for each time series point. This component treats the deviations of the time series that can not be explained by the trend and seasonal component. Figure 2.2 shows a decomposition of a time series (data) in the three components.

2.1.2 Data Discretization

The forecasting methods that are used within the ΘPADx approach and are presented later on in this chapter, all rely on time series as input. As seen in the definition of time series, the measurements are taken in regular time intervals. With other words, the forecasting algorithms require discrete data. However, for real system measurements we can assume that measurements arrive at these discrete time points but it can never be guaranteed. For the context of this thesis, a measurement that is not guaranteed to be discrete is called a *raw measurement*. A sequence of n raw measurements $\{r_1, ..., r_n\}$ within one step size from beginning at t_{start} and end t_{end} can then be defined by:

$$ES_{t_{start},t_{end},M} = \{r_i = (t_i, M, m_i | 0 \leqslant in <, t_{start} \leqslant t_i < t_{end}\}$$

Within the ΘPADx approach and the corresponding implementation the discretization is achieved with various *aggregation methods*. The sequence of raw measurements is then aggregated by the chosen aggregation method. This process is illustrated in Figure 2.3.

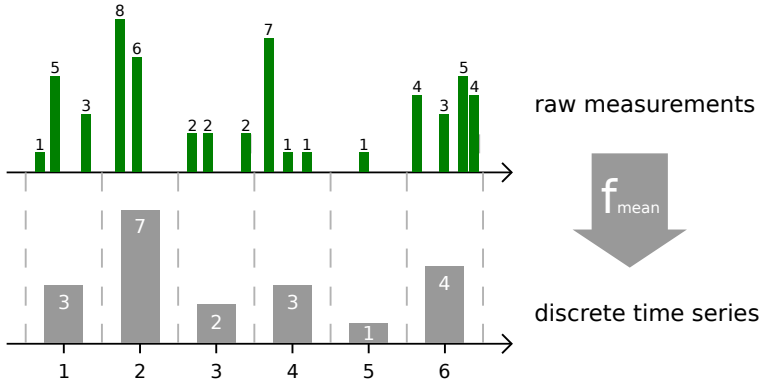

Figure 2.3. The raw measurements are aggregated with the mean function. The result is a discrete time series, that can be processed by the used forecasting methods.

After the execution of the aggregation method a single value is gained from the sequence of raw measurements. The aggregation method can then be described as a function:

$$f : \mathbb{ES} \longrightarrow \mathbb{R}$$

For the aggregation purposes several functions can be used. Simple examples are the sum, min or max function. A full list of all aggregation methods that are considered in the ΘPADx implementation can be found in Table 3.2 of Chapter 3. The most used aggregation function within this thesis is the mean function:

$$f_{mean}(ES) = \frac{\sum_{r_i=(t_i,M,m_i)\in ES} m_i}{n}$$

If for example the mean value of all raw measurements within a minute is used for a graph, this can give a good overview of the nature of the underlying data without smoothing out to much details of the raw data.

2.1.3 Forecasting Methods

The forecasting of a time series value in the future is a common task in a wide range of working fields, for example stock or climate forecasting. Over the time, various forecasting approaches were developed, often highly adapted to the environment they are used in. Examples are forecasting through *Clustering* [Kedia et al. 2005] or with *Neural Networks*

[Zhang et al. 1998]. The forecasting algorithms considered for ΘPADx are working on time series. They are chosen because they rely only on data that can easily be collected from a monitored system. Their calculation is mainly based on the past time series points. Furthermore, all of them are available in the forecasting package of *R* and therefore ready for use. This section first introduces basic terms and afterwards presents common forecasting methods of the time series forecasting domain.

Basic Terms

For the forecasting methods a time series is needed as input. In an online anomaly detection approach, new data is continuously streaming in. Caused by this fact, the processed time series is growing. For every incoming value, the forecast is processed on the growing time series. However, at some point the time series may be too large to process in time or the older time series points are not meant to influence the forecast value any longer. For this purpose the concept of the *sliding window* is introduced [Frank et al. 2001; Shasha and Zhu 2004]. Let $W = \{w_1, ..., w_m\}$ of length $D_W = m$ be a sliding window of a time series X. The sliding window has the same step size $\Delta_W = \Delta_X$ and is of the same length or smaller than X $D_W \leqslant D_X$. In the context of this thesis, a sliding window always holds the most recent m values. This means, if the sliding window already holds m values, and a new value comes in, the oldest value is removed from the sliding window and the actual is added.
The result of a forecasting algorithm is a time series $F = \{f_1, ..., f_l\}$. Each value of this time series is a predicted value of the future, calculated for the same step size that the input sliding window offers. The length $D_F = l$ is called *lead time* [Box and Jenkins 1990] and corresponds to the period of time, predictions should be calculated for.

Moving Average

The moving average method is a simple way to achieve a forecast. Imagine a sliding window $W = \{w_1, ..., w_2\}$ of a time series. The statistical mean function can be used over all values of W. The result of this calculation is the prediction one step size ahead of the last value in W. This process is shown in Figure 2.4. As already mentioned in Section 2.1.2 the applying of the mean function smooths a time series, which may potentially hide outliers. Therefore, the selection of the window size must be chosen with attention.

ARIMA

This forecasting method is based on the selection of a model out of a group of so-called ARIMA models. These models were developed by Box and Jenkins [Box and Jenkins 1990] and consist of three parts:

- AR (Autoregression): Regression operator used for the forecast. This factor influences the weighting of a past time series value for the process of forecasting (see also exponential smoothing).

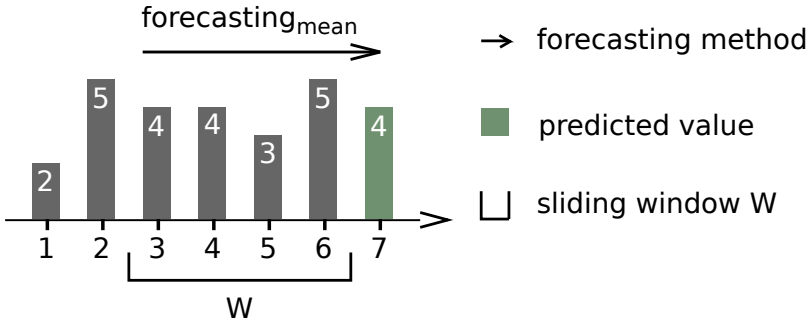

Figure 2.4. Forecasting with the moving average. The predicted value is calculated by the statistical mean function over the sliding window Δ_W. (Figure based on [Bielefeld 2012a])

▻ I (Integration): Integration of the underlying time series to gain a *stationary* time series.

▻ MA (Moving Average): Calculating the average of a sliding window (see Moving Average) of forecast errors. This part covers the assumption, that the values of the time series are interconnected with its forecast errors. Therefore, the forecast errors of past predictions are integrated in the forecasting process.

A time series can be *stationary* or *non-stationary*. With respect to the component model (see Section 2.1.1) a stationary time series has no trend component, while a non-stationary has one. If a prediction for a non-stationary time series is targeted, the time series needs to be differenced to gain a stationary version. Then the AR and MA parts are applied to this time series. Thereafter, the results of the forecast need to be integrated to deal with the non-stationary characteristic. If a forecast for a stationary time series is targeted, the integration part of ARIMA is rejected. Such a model is then called ARMA model.
For forecasts of good quality an ARIMA(p,d,q) model needs to be fitted to a time series. Therefore, p denotes the order of the AR part, q denotes the order of the MA part and d denotes how often the differentiation / integration needs to be processed.

Single Exponential Smoothing (SES)

The single exponential smoothing method sums up weighted values of time series to calculate a future step. The weighting is controlled by a parameter called *smoothing constant*. Let $X = \{x_1, ..., x_n\}$ be a time series, $0 < \alpha < 1$ the smoothing constant and f_t be the forecast for the discrete time point t. This leads to the following formal definition:

$$f_t = \alpha * x_t + (1 - \alpha) * f_{t-1}$$

With the recursion broken down, this can be written as:

$$f_{n+1} = \alpha x_t + \alpha(1 - \alpha)^2 x_{t-1} + \alpha(1 - \alpha)^3 x_{t-2} + ...$$

Within this equation the weighting of the time series points with the help of the smoothing constant α can be seen easily. The youngest time series points have a huge impact on the forecast, while older time series points have lesser impact cause of the exponential scaling of α. In Figure 2.5 the forecasting process is visualized. SES is applied to the sliding window $W = \{4, 4, 3, 5\}$ with a smoothing constant of $\alpha = 0,5$. SES can also be seen as a special case of ARIMA. Only the Integration and the Moving Average parts of an ARIMA model are used (ARIMA(0,1,1)).

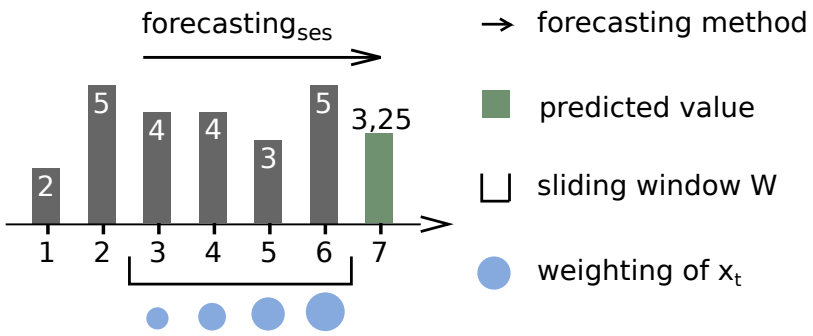

Figure 2.5. Calculating a prediction with single exponential smoothing. Every x_t is weighted by an increasing factor up to $\alpha = 0,5$, visualized by the blue circles. (Figure based on [Bielefeld 2012a])

2.1.4 Anomaly Detection

Anomaly detection is focused on detecting patterns in the data of a system, that do not fit the normal system behavior [Chandula et al. 2009; Steinwart et al. 2005; Lazarevic et al. 2003; Yao 2010]. To achieve this, a so-called reference model is built, that represents the normal behavior of the system. If actual observations significantly deviate from the reference model they can be detected [Yao 2010; Sharma et al. 2003]. These abnormal observations are also called *anomalies* or *outliers*.
Within this section the definitions of anomalies and the reference model are given first. To interpret the deviation of an observation from the reference model, a suiting metric and some kind of border between normal and abnormal behavior is needed. The anomaly score and the anomaly threshold, as defined by Bielefeld [Bielefeld 2012a], are used for this purposes and hence introduced.

Basic Terms

Anomalies are patterns in the data of a system that do not fit to the systems normal behavior. These pattern can have different characteristics, for example they may vary in length or magnitude. Chandola et al. define three different types of anomalies [Chandula et al. 2009].

A *point anomaly* is an individual data point that can be considered abnormal. If a pattern of data can be determined as abnormal in a specific context, but not otherwise, these are called *contextual anomalies*. Finally, if a pattern of related data do not fit the characteristics of the entire data set, these are referred as *collective anomalies*. Figure 2.6 illustrates the three different types of anomalies for time series.

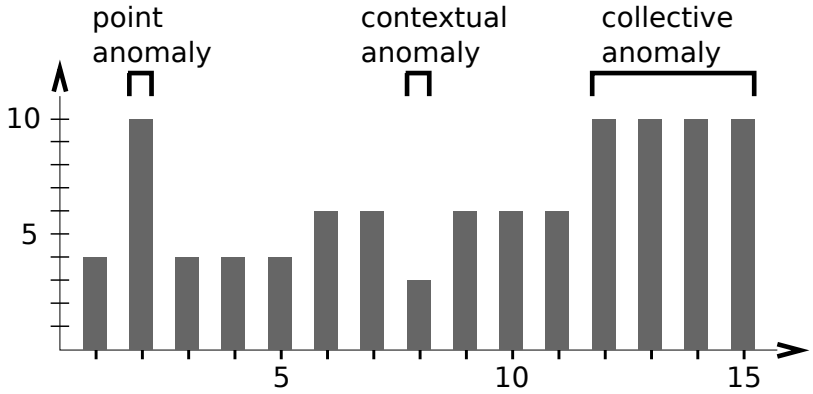

Figure 2.6. Illustration of the three different types of anomalies.

As mentioned, a *reference model* represents the normal behavior of a system. Actual observations are compared to this model to determine if a deviation between them exists. Since this is a fact that strongly influences the quality of the anomaly detection, the search for a fitting reference model is a key challenge. As an example, a modern software system is considered. These are for example often composed of different components or libraries, deployed on different hardware or underlying different workload characteristics. Such parameters influence the behavior of the systems and therefore also the reference model. As a consequence the process of reference model building can not be generalized, but is a highly individual task. The following list shows two common thoughts that form a base for reference model building:

- Many applications have seasonal patterns repeating in regular time periods (see seasonal component in Section 2.1.1). With this knowledge it is possible to get a reference model by expanding the usual appearance of the pattern into the future.

- For most applications the following assumption can be made. Actual monitoring values measured, for example response times or database connections, may not differ much from the last measured values. For example, the introduced forecasting algorithms are based on this assumption, since they all use the past time series values to calculate a prediction.

Anomaly Score and Anomaly Threshold

An anomaly score should be able to express the difference between the expected value from the reference model and the actual measured value. May X be the time series of incoming measurements from a monitored system and Y the time series of the corresponding reference model. As base of the anomaly score the euclidean distance of two time series points x_i and y_i of X and Y at time i can be calculated by:

$$D_{euclid}(x_i, y_i) = \sqrt{(x_i - y_i)^2}$$

This results in a value that represents the distance between the two time series points. A normalization is applied to gain a value that is easy to interpret and comparable to values that are generated by other time series:

$$A(x_i, y_i) = \left| \frac{D(x_i, y_i)}{x_i + y_i} \right| \in [0, 1]$$

The result of the normalization is the so-called anomaly score, which offers values between 0 and 1. ΘPADx is able to process anomaly detection for multiple applications at the same time. With the defined anomaly score, it is possible to compare all received anomaly detection results. This will help to find the cause of an anomaly within a complex software system, as will be shown in the evaluation (see Chapter 5).
It is now possible to define a value $\theta \in [0, 1]$, called *anomaly threshold*. Whenever the anomaly score exceeds this value, the corresponding measurement is classified as anomaly.

2.2 Anomaly Detection Quality Metrics

There can be different scenarios for anomaly detection approaches. For example, an anomaly can be detected if an anomaly occurs. This would be a correct warning. But it is also possible to detect an anomaly even if none exists, which would be a false warning. There are two more cases, a missing warning and correctly detected occurrence of no anomaly. As pointed out by Salfner et al. [Salfner et al. 2010] this results in four different cases, that are summed up in Table 2.1. According to Salfner et al. [Salfner et al. 2010] these cases lead to important metrics, that are used to compare the quality of detection results. In addition to these metrics, ROC curves are introduced. These are used to visualize the quality of detection results.

2.2.1 Detection Comparison Metrics

In this section the metrics *true positive rate*, *false positive rate*, *precision*, *accuracy* and the *F-measure* are introduced. They are all derived from the mentioned four cases that are shown in Table 2.1.

True Positive Rate (TPR)

If an anomaly occurs and an algorithm is detecting it correctly, this is called a *true positive* (TP). It could also be the case, that an anomaly occurs but the algorithm signals a normal value, a so-called *false negative* (FN). The sum of those two values represents the number of all occurred anomalies. If this is set in relation to the true positives, the resulting value is called *true positive rate* (TPR):

$$true\ positive\ rate = \frac{TP}{TP + FN} = \frac{TP}{F}$$

False Positive Rate (FPR)

An anomaly detection algorithm can also assume that an anomaly occurred, even if the actual behavior is normal. This case is called a *false positive* (FP). For the *false positive rate* (FPR) these false warnings are related to the false positives and *true negatives* (TN). Where a true negative is the case when the actual behavior is normal and the anomaly detection also assumes a normal behavior. This is basically the ratio between false warnings and all measures that were no anomalies:

$$false\ positive\ rate = \frac{FP}{FP + TN} = \frac{FP}{NF}$$

Precision

After a detection exists a number of detected anomalies (POS). The *precision* is the ratio between the correct detected anomalies (TP) and all detected anomalies:

$$precision = \frac{TP}{TP + FP} = \frac{TP}{POS}$$

Accuracy

The *accuracy* is the ratio of all correct detections to the number of all detections carried out:

$$accuracy = \frac{TP + TN}{N} = \frac{TP + TN}{TP + FP + FN + TN}$$

F-Measure

If the precision of an approach is increased, it is often accompanied by the true positive rate getting worse. To deal with this trade-off, Rijsbergen [Rijsbergen 1979] introduced the *F-measure* as combination of precision and true positive rate:

$$F - measure = \frac{2 \cdot precision \cdot TPR}{precision + TPR} \in [0, 1]$$

The higher the quality of an approach is, the higher is the result of the F-measure [Salfner et al. 2010].

Table 2.1. This table shows all possible cases for the outcome of a detection. (Based on [Salfner et al. 2010])

Detection Contingency Table			
	True Failure	**True Non-failure**	**Sum**
Prediction: Failure	true positive (TP) (correct warning)	false positive (FP) (false warning)	positives (POS)
Prediction: No failure	false negative (FN) (missing warning)	true negative (TN) (correctly no warning)	negatives (NEG)
Sum	failures (F)	non-failures (NF)	total (N)

In Chapter 5 these metrics are used to compare the different configurations of the ΘPADx approach and to assess the quality of the anomaly detection results. Therefore a large set of data is collected. This set is revisited in form of a manual anomaly detection. The intervals within anomalies occurred are marked. It is then possible to compare the marked intervals with the results of the ΘPADx approach and to derive the four different cases showed in Table 2.1. Finally the detection comparison metrics can then be calculated and used for comparison.

2.2.2 ROC Curves

Receiver operating characteristic (ROC) curves are a common way to visualize the performance of detection approaches [Maxion and Roberts 2004]. As already mentioned in the context of the F-measure, the true positive rate and the false positive rate are changing in relation to each other, depending on a varying threshold. A ROC curve is a two-dimensional graph that depicts these two values of a detection approach against each other and is therefore able to visualize this tradeoff [Maxion and Roberts 2004; Fawcett 2006]. Figure 2.7 shows two ROC curves, illustrating curves for high and low accuracy and the dependence to varying thresholds. The ROC curves will help to compare the results of the ΘPADx approach under certain configurations against each other. This is necessary for the evaluation, for example to find the best possible configuration of ΘPADx for a given time series.

2.3 Performance Metrics

Measurements can be made under certain points of view called *metrics* [Rezagholi 2004]. For example, the number of function calls, the execution time of a function, or the memory consumption can be measured. The monitoring tools used by Xing are able to measure various metrics for the running applications. Hence the used metrics are listed:

- **Response Time** The response time, interpreted as the time between the user finished the request until the system completes the response, as for example introduced in [Koziolek

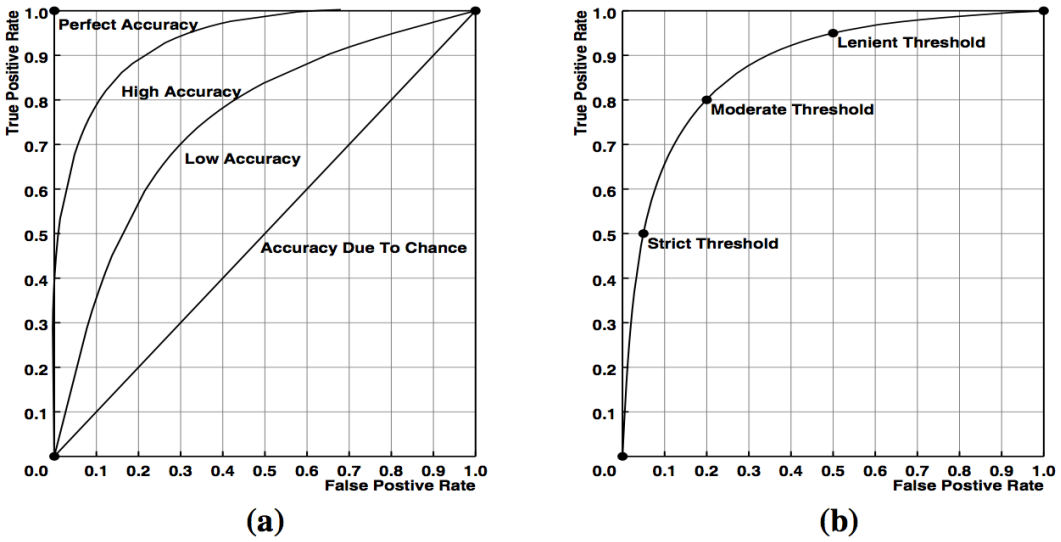

Figure 2.7. (a) shows several curves from low to high accuracy. A perfect detection approach is located in the upper left corner while a random guess approach lies on the diagonal at the center. (b) illustrates the dependency between the outcome of a detection approach and the used threshold. (Figure from [Maxion and Roberts 2004])

2008]. Within the Xing monitoring environment, the response time is called *total time* and concludes request times, which are needed for requesting other components, for example the database or the memcache.

- **Requests** The amount of requests to the system in a given time span. For example the amount of requests to a database per second.
- **Database Time and Database Calls** The average response time for the database calls per request is named *database time* and the amount of requests to the database in a given time span is called *database calls*.
- **Search Time and Search Calls** The average response time for the search calls per request is named *search time*. The amount of calls to the search API is named *search calls*.
- **Memcache Time and Memcache Calls** Xing is using a memcache, that stores often used data or data that must be retrieved from a database or an API. This can increase the performance, since database or API calls can be expensive in case of performance. The average response time for the memcache calls per request is named *memcache time*. The corresponding amount of calls to the memcache is named *memcache calls*.
- **Rest Time and Rest Calls** The average response time for a call to the rest API per request is named *rest time*, while the corresponding amount of calls to the rest API in a

given time is named *rest calls*.

- **Gearman Time and Gearman Calls** Xing uses the job server Gearman[1]. With Gearman, it is possible to source out certain jobs to other machines or processes. Therefore, it can be used to work in parallel or for load balancing purposes. The average response time for a gearman call per request is named *gearman time*. The amount of gearman calls is in a given time span is named *gearman calls*.

- **Other Time** At Xing, the *other time* contains everything that is not covered by the database, search, memcache or gearman time, but still influences the total time.

2.4 Tools and Technologies

The following chapter lists the most important tools and technologies that are used for the implementation of the ΘPADx approach or are part of the evaluation environment. Since all tools and technologies are based on several protocols and concepts, these are introduced in a first step, followed by the tools and technologies itself. ΘPADx is evaluated in the Xing environment, therefore the presentation is separated into two parts. The first part describes *Kieker*, *R* and *MongoDB*, which are used for the implementation of ΘPADx. The second part introduces the most important tools and technologies used within the Xing monitoring infrastructure.

2.4.1 Protocols and Concepts

ΘPADx is able to communicate with its environment, for example to gather data from a monitored system or to submit its results to an alert-facility. Over the last years some new approaches for communication purposes evolved, still using the common protocol stack of the internet. The focus lies on new data formats, that are readable for machines and humans. Corresponding protocols and concepts are presented in the following.

JSON

The JavaScript Object Notation (JSON) is a text-based data interchange format and therefore used for the serialization of structured data. It is able to represent four primitive types (strings, numbers, booleans and null). Additionally, two structured types can be represented (objects and arrays). Data can be nested arbitrarily. In difference to, e.g., XML, JSON is more lightweight and has a smaller syntactical overhead, which ensures a better human readability.

[1] http://www.gearman.org

BSON

Binary JSON[2] (BSON) is a binary-encoded serialization of simple documents, in particular JSON documents. Since it has small storage overhead and can be traversed easily, it is used for data representation within MongoDB, a noSQL data base that will be described later on in this chapter.

NoSQL-Databases

Within the last years a new kind of databases, the so-called *NoSQL* databases are gaining more and more attention. These break the long term dominance of databases that pursues the *relational* approach. Relational databases are organized and structured with respect to a relational model, containing for example tables and relations between them. The NoSQL style is more focused on high scalability and flexibility, but provides less functionality. Therefore, these kinds of databases are predestined for storing huge quantities of data while offering fast access times for simple retrieval concerns. A subgroup of NoSQL databases is formed by the so-called *document store databases*. Here, the stored data is text-based and relies on standard formats like the prior mentioned JSON. Such *documents* are organized and grouped in different ways. They can be hold in *collections* or can be identified via *tags* for example.

2.4.2 Tools and Technologies of the ΘPADx implementation

The ΘPADx implementation uses several tools and technologies to gain useful structural properties or functionality. These are introduced within this section.

Kieker Monitoring Framework

The Software Engineering Group at Kiel University developed *Kieker* [3], a flexible framework for monitoring and analyzing purposes. It is able to work on concurrent or distributed software systems. Through its architecture, Kieker produces only a small amount of overhead even in high scalable systems [van Hoorn et al. 2009]. As seen in Figure 2.8, Kieker provides a Monitoring Component, that includes so-called Monitoring Probes. After being injected into the monitored system, these probes are able to collect measurements, which are stored and communicated to the Analysis Component as Monitoring Records. Within the Analysis Component, the needed records are read and passed to Analysis Plugins, that are connected in pipes-and-filters style. This architecture makes the analysis solution highly flexible and configurable. It is possible to use the supplied Kieker Analysis Plugins or to easily write your own plugins. Because of the pipes-and-filters pattern, it is easy to exchange filters or add additional filters to the analyzing process [van Hoorn et al. 2012].

[2] http://bsonspec.org
[3] Official web site of the Kieker framework: `http://kieker-monitoring.net/`

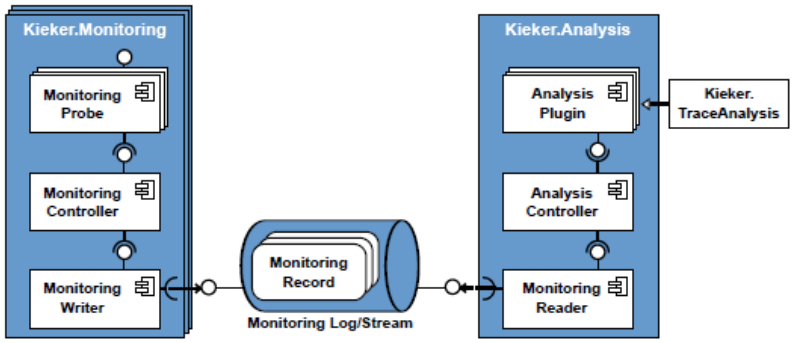

Figure 2.8. Architecture of the Kieker Framework [Kieker Project 2013]

Kieker is the base of the upcoming implementation, favored due to its extensibility and flexibility. Because of the refactoring of the ΘPAD approach to a pipes-and-filters structure, the extended approach can be achieved by adding new self-written filters to the ΘPAD analysis configuration or by modification of already available ΘPAD filters. The components of ΘPAD that are reused in ΘPADx, the new filters and the modifications to existing filters are presented in detail in Chapter 4.

The *R* Project

R [R] is a programming language specially designed for statistical computing and graphics. It is based on the *S* language, which was developed to handle the growing amount of statistical computing at the Bell Laboratories (formerly AT&T, now Lucent Technologies) [Bell 1994]. Unlike *S*, *R* is an open source project under GPL license. It is flexible and extensible, whereby it supports a wide variety of statistical methods. Important for this thesis is the forecast package [Hyndman 2013], which is available over the CRAN[4] package repository. This package includes all the forecasting algorithms, that are used in the here presented process of anomaly detection, e.g. ARIMA or Exponential Smoothing.

MongoDB

For the ΘPADx approach, the storage of huge data sets is necessary. Time series points representing data from different applications, under different metrics and over a long time period must be stored. Given these facts, the NoSQL concept provides the solution of choice for ΘPADx. MongoDB is chosen in particular because it is the leading NoSQL database. Furthermore, it is open source and offers a well documented API. MongoDB also has an easy to use Java driver, which simplifies the connection to ΘPADx, that will be

[4] http://cran.r-project.org/web/packages/forecast/

developed in the Java[5] programming language. Besides this, several MongoDB instances are already in use at Xing. This brings existing technical knowledge and low integration barriers with it.

2.4.3 Xing - Case Study Environment

Xing is a social network system hosted by the Xing AG. It is Europe's most important network aiming on business contacts[6]. Xing originally started under the brand name openBC in 2003. Its fast growing user base leads to a rework of the used architecture and to renaming it Xing in 2006. Users are able to share their profiles, create groups, or organize events via Xing to search and find expert knowledge, jobs, employees, or ideas [Xing AG 2012].

To handle the high amount of users, currently about 13 million users world wide [Xing AG 2013b], Xing owns hundreds of servers. To store the users and their content, several SQL servers are running. The majority of the servers is used to run the applications, which together build the Xing Network. Most of them are written in Ruby on Rails and Perl. In Table 2.2 the here considered applications and their communication behavior can be seen. Given the huge amount of servers and users, a strong communication tier exists. While *HTTP REST* is used for synchronous communication, the asynchronous communication is achieved with *ZeroMQ*. Several monitoring tools are listening on the communication tier to collect the needed data and to keep watch on the behavior of the system. Besides tools like Munin[7], Nagios[8], or Omniture[9], two self-written applications are in use: Logjam and Graylog2.

Ruby on Rails

Ruby on Rails [Rails] is an open source Web Application Framework written in Ruby. It was presented for the first time in July 2004. Rails is made for a fast development process and has a large supporting community. The core of Rails follows the *MVC* pattern. Communication is achieved via *REST*.

Perl

Perl [Perl] was developed in 1987 and is therefore a long established scripting language. It is feature-rich, runs on a huge amount of platforms, and offers a huge amount of third party modules. In addition to its flexibility, Perl is especially known for its powerful text processing and manipulation facilities.

[5] http://www.java.com
[6] http://corporate.xing.com/index.php?id=108&L=0&tx_ttnews[tt_news]=774
[7] http://munin-monitoring.org/
[8] http://www.nagios.org/
[9] http://www.omniture.com/

Table 2.2. This table shows how the considered applications are calling each other. This can help to correlate the anomaly detection results of applications that are interconnected.

Caller	Callee	Caller	Callee
bestoffers	perlrest	perlyaml	companies
bestoffers	perlyaml	perlyaml	events
		perlyaml	jobs
communities	perlrest	perlyaml	perlrest
		perlyaml	profiles
companies	jobs	perlyaml	riaktivities
companies	perlrest		
companies	perlyaml	profiles	events
companies	riaktivities	profiles	perlrest
companies	triton		
		publicsearch	perlrest
events	companies	publicsearch	triton
events	perlrest		
events	perlyaml	riaktivities	companies
events	triton	riaktivities	events
		riaktivities	jobs
ipad	jobs	riaktivities	perlrest
ipad	perlrest		
ipad	riaktivities	perlapp	companies
ipad	triton	perlapp	events
		perlapp	jobs
jobs	companies	perlapp	perlrest
jobs	perlrest	perlapp	perlyaml
jobs	triton	perlapp	riaktivities
		perlapp	triton
messages	perlrest		
perlapp	companies		
perlapp	events		
perlapp	jobs		
perlapp	perlrest		
perlapp	profiles		
perlapp	riaktivities		
perlapp	triton		
perlrest	companies		
perlrest	events		
perlrest	jobs		
perlrest	perlrest		
perlrest	profiles		
perlrest	riaktivities		

ZeroMQ

ZeroMQ [ZeroMQ] is an asynchronous messaging library, designed for use in distributed or concurrent applications. It supports more than 30 languages and common operating systems. Unlike other messaging middlewares, ZeroMQ can run without a dedicated message broker. It provides a socket-like API, used to build a message queue with certain supported patterns, e.g. Request-Reply, Publish-Subscribe, or Fan-in / Fan-out.

Logjam

The logs and data produced by several applications and servers at the Xing environment are written in the message queue powered by ZeroMQ. The information is aggregated and stored in a database by a server, the so-called *Importer*. This information is corresponding to the already mentioned metrics and can be used for debugging and monitoring purposes. To support these processes, a web front-end was developed by Dr. Stefan Kaes for the import of the stored information and its visualization as tables and graphs. Logjam [Kaes 2013] is able to show a live view of the incoming data, but is also able to browse the historical data, as demonstrated by Bielefeld [Bielefeld 2012a]. Figure 2.9 shows an overview of the here described process of monitoring at Xing.

Graylog2

Graylog2 [GrayLog2] is an open source log management tool. It has a server written in Java, that accepts log messages via certain messaging protocols and stores them into ElasticSearch[10]. Finally, the collected data is presented and can be managed by a web interface.

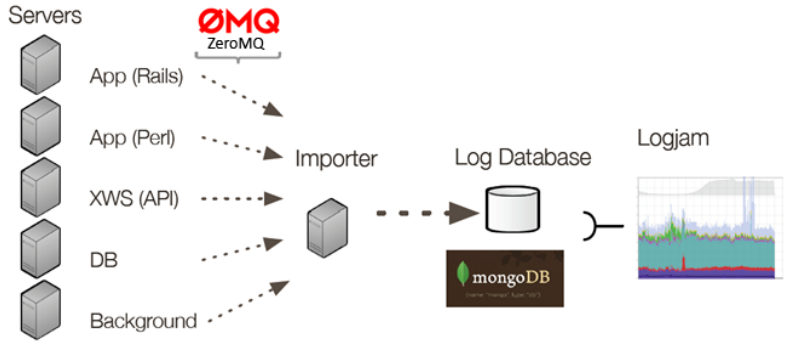

Figure 2.9. An overview of the Xing monitoring architecture. (Image based on [Bielefeld 2012a])

[10]http://www.elasticsearch.org/

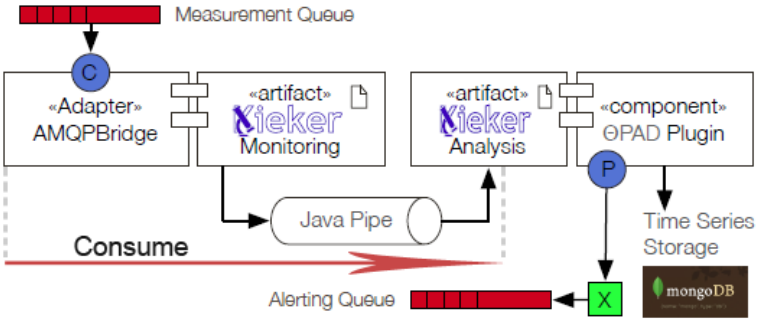

Figure 2.10. High-level Architecture of the ΘPAD approach with the Consume step marked [Bielefeld 2012a].

2.5 The ΘPAD Approach

The Online Performance Anomaly Detection (ΘPAD) approach is a concept developed by Tillmann Carlos Bielefeld [Bielefeld 2012a]. The ΘPAD approach is introduced because the ΘPADx approach is based on it. With this introduction it is easier to identify and understand the extensions that are presented in the upcoming Chapters 3 and 4. ΘPAD uses forecasting algorithms on time series to create a reference model, based on the data of a monitored system. This model is compared to real time measurements of the system. The distance between the values of the reference model and the real measurements are normalized and interpreted. If they exceed a given threshold, an anomaly is detected. The process of anomaly detection in the ΘPAD approach can be divided in several steps, as depicted in Figures 2.10 and 2.11. A brief introduction to each step is given in the remainder of this section:

1. Consume
The monitored system passes its information for example to a provided AMQP[11] based messaging queue. For ΘPAD the information of interest are the response times of a single application. The Kieker Monitoring Component is used to consume the data from the queue and transform the data into Monitoring Records. The records are eventually passed to the Kieker Analysis Component.

2. Reader
The Kieker Analysis Component is able to read the records with a so-called Monitoring Reader. The information is passed to the ΘPAD plugin, that is also placed in the Kieker Analysis Component.

[11] http://www.amqp.org

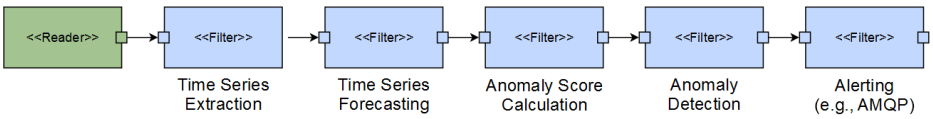

Figure 2.11. The several steps of ΘPAD restructured to fit the new Kieker pipes-and-filters architecture [Bielefeld 2012b].

3. Time Series Extraction
The plugin continuously consumes measurements of the monitored system. The measurements mostly arrive in different time intervals and are therefore needed to be discretized. The incoming data is aggregated over a configurable time span with the mean function as aggregation method. The result of this step is a time series as defined in Chapter 2, ready to be passed on to the next step.

4. Time Series Forecasting
In this step, the next value of the time series is forecasted with one of the presented forecasting algorithms (see Chapter 2). The calculations are based on the previous values of the time series within a configurable window size. This process builds the reference model, as defined in Chapter 2.

5. Anomaly Score Calculation
The forecasted value is compared to its actual measurement as soon as it arrives. With the metric presented in Chapter 2, a distance between these points can be calculated. The calculated distance is normalized to a value between 0 and 1, the result is the anomaly score (see Section 2.1.4).

6. Anomaly Detection
To detect an anomaly a configurable threshold value is used, the anomaly threshold (see Section 2.1.4). This value acts like a border between the normal system behavior and an anomaly. If the calculated anomaly score exceeds the threshold, an anomaly is detected by the ΘPAD approach. As you can see, a decent configuration of the threshold is necessary to achieve a useful anomaly detection.

7. Alerting
If an anomaly is detected, an alert is reported to the connected alerting queue.

Chapter 3

The ΘPADx Approach

In this chapter, the process of anomaly detection with ΘPADx is discussed in a more abstract manner before the implementation details are described in the next chapter. All the activities, that together build up the anomaly detection process, are discussed and the interaction between them is shown. To achieve this, an overview of all activities is given first, followed by a detailed look on each activity on its own. Some of the activities are similar to the steps of the ΘPAD approach as shown in the last chapter. For those only a short summary of their known functionality is given and the focus remains on the differences.

3.1 ΘPADx Approach Overview

ΘPADx stands for Extended Online Performance Anomaly Detection. As the name indicates, ΘPADx is designed to detect anomalies in performance data, for example response times or workload intensities of a monitored software system. Since ΘPADx is an approach to process anomaly detection online, it must be able to process data that continuously streams in at runtime. This is a huge difference in contrast to offline approaches, because in the case of offline anomaly detection, the complete set of data to process is already known. ΘPADx is based on ΘPAD that was developed by Tillmann Bielefeld in 2012 [Bielefeld 2012a] and briefly introduced in the last chapter. It produces anomaly detection results of good quality especially for contextual anomalies. The approach however is still facing problems with collective anomalies, which negatively affects the precision of the anomaly detection. Also, an ΘPAD instance is only able to process performance data of a specific performance metric from a single application. To gain a better coverage of a software system in case of anomaly detection, the possibility to process multiple metrics of multiple applications per instance is necessary. Several extensions and changes to ΘPAD are made to deal with these issues. These are explained in the detailed presentation of the corresponding activities.

Figure 3.1 shows all activities, which are necessary to set up and process the ΘPADx approach. They are divided in two major parts, offline and online activities. There is only one offline activity, the configuration (A1). Since ΘPADx uses for example forecasting algorithms and discretization, suitable methods and corresponding parameters must be

chosen to gain results of high quality. The configuration is also accountable for the selection of the applications and corresponding metrics for which an anomaly detection should be processed. In addition, there are three online activities that are executed for every incoming measurement. As a first activity, the *data extraction* (B1) takes place, where the incoming measurements are parsed to time series points. With the help of the forecasting algorithms, which can now process the time series data, the anomaly score is calculated and interpreted (B2). This activity equals the approach, which was chosen by Bielefeld within ΘPAD as most important part of the anomaly detection. For ΘPADx this activity offers the most extensive changes in contrast to ΘPAD. The forecasting process is adjusted and an additional forecaster is introduced, the so-called *pattern checking*. This forecaster makes use of a long-term collection of time series data to improve the anomaly detection in case of the appearance of collective anomalies. The last activity is the *send and store activity* (B3). Since it is the last activity in the process, it is responsible for the storage of the detection results and the forwarding of the results to connected alarm facilities. An activity can contain sub-activities, which are called *steps* within this thesis.

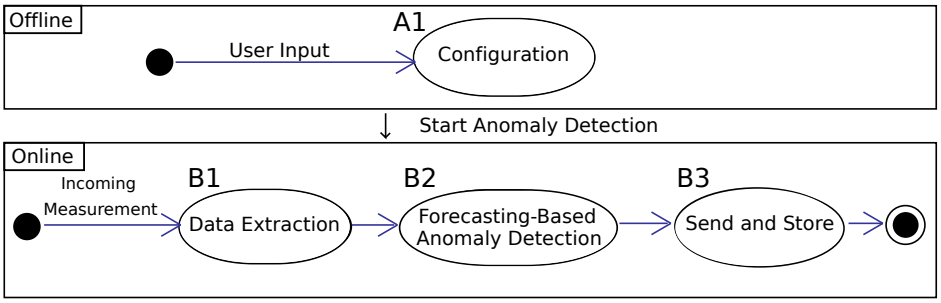

Figure 3.1. Activities of the ΘPADx approach. After the Configuration (A1), the anomaly detection can be started. First the incoming measurements are transformed to time series data (B1), then the anomaly score is calculated and interpreted (B2). Finally the results are stored and sent to connected facilities (B3).

The activities *forecasting-based anomaly detection* and the additional forecaster *pattern checking* are considered the core of the anomaly detection process. Within ΘPAD the configured forecasting method is used in every situation. For ΘPADx this behavior is changed: Whenever the approach faces a collective anomaly, the pattern checking forecasting method is preferred. Instead of a one part approach, now a hybrid approach of two forecasting methods is chosen. Since the adjusted forecasting and the pattern checking is the main extension to ΘPAD, a huge part of the next section will be used to explain this activity in detail.

3.2 ΘPADx Approach Activities

3.2.1 Configuration (A1)

Bielefeld [Bielefeld 2012a] introduced the concept of aspects within his thesis. An aspect is a separated unit of measure and analysis. It holds all the needed parameters for an anomaly detection. But with one aspect, only one metric of an application can be processed. In a large software system often multiple applications exist. These applications can offer a huge load of information. For example the response times, the amount of database connections, the amount of current users, or other metrics, as defined in Chapter 2.3. Therefore, it is necessary to start more and more instances of ΘPAD with configured aspects for a growing amount of applications and metrics that should be processed with ΘPAD. This leads to an unnecessary overhead, that should be avoided within the ΘPADx approach.

ΘPADx is capable of handling multiple applications with multiple metrics in a single instance. The name of an application and a metric are building an *identifier*. Every online activity of ΘPADx is then able to separate the incoming data based on the identifiers. This keeps the anomaly detection of several applications and metrics uninfluenced by each other. For this chapter, the assumption is made, that all activities and steps are able to process this separation. How the separation is accomplished in the implementation, is then shown in the next chapter. With this opportunity only a single instance of ΘPADx is necessary to process the anomaly detection for multiple applications and metrics. ΘPADx accepts a list of application-metric pairs as parameters, an example list is given in Table 3.1. The details of the implementation changes to apply the identifier concept and the correct separation of the different identifiers are discussed in Chapter 4.

Within the configuration activity, also parameters for the discretization (see Section 2.1.2) can be chosen. This includes an aggregation method and the time span over which the aggregation will appear. This time span corresponds to the step size that the resulting discrete time series will have.

ΘPADx supports different forecasting algorithms to calculate a reference model. Within the configuration process, one of these algorithms can be chosen. The forecasting algorithms are working on time series. ΘPADx is developed to achieve anomaly detection on the performance data of software systems. Depending on different factors like workload or underlying hardware, these time series can have different characteristics. As for example stated by Chan and Mahoney [Chan and Mahoney 2005], it is necessary to select a forecasting algorithm that fits the characteristics of the time series produced by the software system. Otherwise, the results of the forecasting and therefore also the results of the anomaly detection can be of poor quality. It follows, that a deeper knowledge of the targeted software system and the available forecasting algorithms is a prerequisite for the practical usage of ΘPADx.

Table 3.1. This table shows all parameters that can be configured for ΘPADx.

ΘPADx Configuration Parameters	
Identifier Parameters	
Name	Example
Application-Metric List	{mainApplication.db_time, backgroundProcess.total_time}
Aggregation Parameters	
Name	Example
Aggregation Time Span	5 minutes, 30 seconds
Aggregation Method	SUM, MAX, MEAN
Time Series Analysis Parameters	
Name	Example
Forecasting Algorithm	ARIMA101, MEAN
Forecasting Time Window Capacity	10, 60, 100
Extended Approach Parameters	
Name	Example
Pattern Checking active	true, false
Consecutive Anomalies Threshold	2, 5, 10
Distance	1 Day, 1 Week, 12 Hours
Iteration	1, 2, 10
Anomaly Interpretation Parameters	
Name	Example
Threshold	0.5, 0.23, 0.9

As already mentioned, the appearance of collective anomalies affect the quality of the anomaly detection negatively. With ΘPADx an additional forecaster can be used within the anomaly score calculation activity to deal with this issue. It is only applied, if a configurable number of consecutive anomalies is already detected, which indicates a collective anomaly. This step is called *Pattern Checking* and will be presented in detail in Section 3.2.3. Due to the structure of Kieker plugins, it can easily be activated or deactivated. Of course this option is also part of the configuration. If activated, some additional parameters for this step are necessary. The pattern check forecaster compares the actual time series window with a time series window in the past that matches an corresponding seasonal pattern, as introduced in the component model (see Section 2.1.1). For this purpose, a distance can be given as parameter, that represents the length of a seasonal pattern. It is also possible to go

not only one seasonal interval in the past, but to iterate over several intervals that match the seasonal pattern. Therefore, also a parameter can be chosen to determine how often this repetition should occur.

For the interpretation of the calculated anomaly scores, an anomaly threshold is used, as introduced in Section 2.1.4. This threshold can also be configured within the configuration activity.

All mentioned parameters are summarized in Table 3.1 to grant a complete overview of an ΘPADx configuration. The configuration is a key activity to gain anomaly detection results of high quality. As already stated, knowledge and experience of the underlying system and the forecasting algorithms are helpful to determine suitable parameters. This is a common fact that can often be found in the literature. For example Oliner and Stearley [Oliner and Stearley 2007] investigated the logs of super computers. They were facing a huge amount of data, in their case millions of log files. A key observation was that the interpretation of the data is highly interrelated with so-called *operational context*. Among other factors, this also includes expert knowledge of the underlying system.

3.2.2 Time Series Extraction (B1)

The time series extraction is the first activity that is processed online. It creates a time series as defined in Section 2.1.1 from a stream of Kieker Monitoring Records. This is achieved by executing two consecutive steps, the data extraction and the discretization. Figure 3.2 gives an overview of the time series extraction activity and the containing steps in the process of anomaly detection. In the following the steps of the time series extraction are explained.

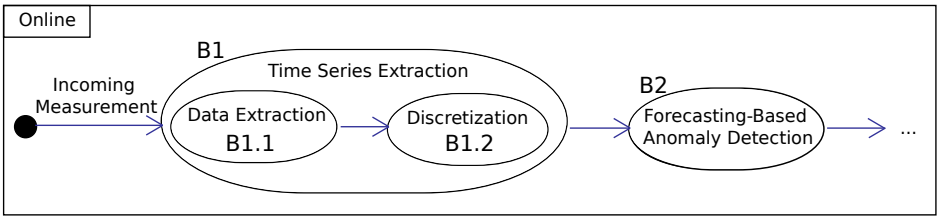

Figure 3.2. Overview of the Time Series Extraction activity (B1). It contains two steps: Data Extraction (B1.1) and Discretization (B1.2).

Data Extraction (B1.1)

A Kieker Monitoring Record represents the incoming measurements and contains information like an identifier, a timestamp, and a measured value. It is created as soon as the raw data arrives. The record is sent to the ΘPADx Analysis component. The details of the

Table 3.2. All supported aggregation methods of ΘPADx.

Aggregation Methods	
Aggregation Method	ΘPADx Parameter
Geometric Mean	GEOMETRIC_MEAN
Maximum	MAX
Mean	MEAN
Minimum	MIN
Percentile	PERCENTILE90 or PERCENTILE95
Product	PRODUCT
Sum	SUM
Sum of the Squares	SUMSQ
Sum of natural Logs	SUMLOG
Variance	VARIANCE

sending process are explained in the next chapter. First, the record is processed by the data extraction. The mentioned information is extracted from the record and forms a time series point. Within ΘPADx a time series point is also a record that can be further processed by the other activities. A measurement however, and therefore also a record, can arrive at any time. A combination of the extracted time series points would then obviously not build a discrete time series. To accomplish this, the discretization is processed after the data extraction.

Discretization (B1.2)

In this step, the discretization, as introduced in Section 2.1.2, is applied. The incoming time series points are collected over a configurable time span (see Table 3.1) that correspond to the step size Δ_X of the resulting discrete time series X. The timestamp of the first incoming time series point sets the start time point t_{start}. The end time point t_{end} is then calculated by:

$$t_{end} = t_{start} + \Delta_X$$

The following incoming time series points are collected. Whenever a time series point, whose timestamp exceeds t_{end}, is received, the collection is closed and an aggregation method is applied. ΘPADx offers different aggregation methods that can be selected within A1 and are shown in Table 3.2. For example the mean or the maximal value of all collected values can be calculated. The result of the aggregation process is one time series point per aggregation time span. More of these time series points build up to a discrete time series. As already explained, discrete time series are a prerequisite for the usage of the forecasting algorithms.

3.2.3 Forecasting-Based Anomaly Detection (B2)

This activity describes the main part to accomplish the anomaly detection of the approach that was already used by Bielefeld within ΘPAD and is now extended. The main idea is to compare a reference model generated by forecasting methods with actual measurements. The derivation between them is used to calculate an anomaly score, that describes the level of abnormality. Finally, the anomaly score is interpreted with the help of an anomaly threshold. However, the results of Bielefeld [Bielefeld 2012a] reveal, that the performance of the used forecasting methods is not as good as desired. This behavior is analyzed in the following and the corresponding improvements are introduced. After that, it is shown how the improvements are included in the forecasting-based anomaly detection activity. A first overview over the activity is given in Figure 3.3.

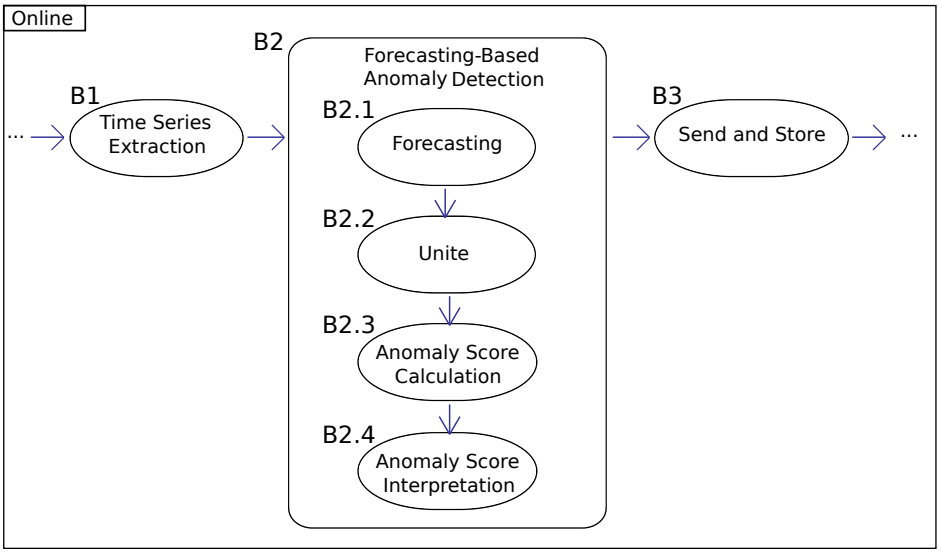

Figure 3.3. Overview of the forecasting-based anomaly detection activity (B2). First a forecasting appears (B2.1) the results are used as reference model. The forecasts are united with the corresponding actual time series point (B2.2). Then the anomaly score is calculated (B2.3) and finally interpreted (B2.4).

ΘPAD Anomaly Detection Quality Analysis

The pattern checking forecaster is an extension to the original ΘPAD approach. It is based on the fact, that time series have a seasonal pattern, as defined by the component model (see Section 2.1.1). These pattern are used to define an alternative reference model. Based on the alternative reference model, an additional anomaly detection can be performed. This section will show why this additional activity is necessary to improve the quality of

the anomaly detection results of the original ΘPAD approach.

Bielefeld executed an extensive evaluation phase within his thesis [Bielefeld 2012a]. With the best found configuration of ΘPAD, the approach reached an accuracy of 0.97 and a precision of 0.13. The high accuracy implies that, if ΘPAD signals an anomaly, the probability that this actually is an anomaly is high [Bielefeld 2012a, p. 86]. However, the quality of the anomaly detection is negatively influenced by the low precision value. It implies that it is likely, that the approach ignores occurring real anomalies.

Within the evaluation process of Bielefeld, the mean forecaster was the best fit on the evaluation data and was used in the best found configuration. Time series analysis based forecasting methods, like the mean forecaster, have several advantages. They are easy to use, easy to understand, fast and only need a small amount of data to calculate a forecast. However, as stated by Sharma et al. [Sharma et al. 2003], these methods are a good choice for point anomalies, but are not suited for collective anomalies. This fact also matches the results of Bielefelds further analysis of his evaluation data. Figure 3.4 demonstrates this problem with the help of example time series. The grey bar graph shows the time series of the measurements, including an collective anomaly. The green graph shows the corresponding forecasts accomplished by the mean forecaster and the red graph shows the calculated anomaly score over the time. As soon as the collective anomaly starts, the anomaly detection shows an increased anomaly score (1). The start of the collective anomaly is detected properly. But when the anomaly continues over a longer time period, the sliding window of the underlying mean forecaster is filled with the now abnormal values. As soon as the collective anomaly is over, the measured values and the anomaly score return to the normal level (2). Now there is again a deviation between the values of the sliding window and the actual measurements. This again results in an increasing anomaly score (3) and, if high enough, in a second detected anomaly, even if there was just one collective anomaly. This behavior also explains the low precision of the approach. The values at (3) may exceed the configured anomaly threshold and therefore cause false positives. Since the false positives are used to calculate the precision (see Section 2.2.1), this lowers the result.

Since the introduced problem appears only if a collective anomaly is processed, the pattern checking is only needed in this case. Therefore, the consecutive anomalies detected must be counted. If they exceed a configurable threshold, it is assumed that the approach faces a collective anomaly.

Pattern Checking Forecaster

In the previous section, it was shown why the ΘPAD approach produces additional false positives. Within this section, the pattern checking forecaster is introduced, which aims at preventing this behavior. The pattern checking forecaster creates an alternative reference

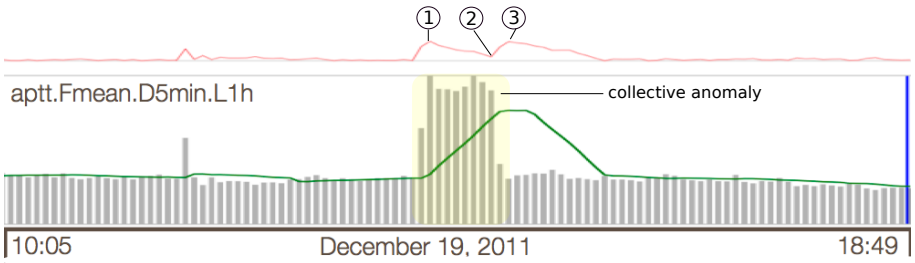

Figure 3.4. This figure shows the cause of the low precision of the ΘPAD approach with the help of an example. The approach first detects the starting collective anomaly correctly (1). It then adapts to the abnormal measurements and interprets them as normal (2). As soon as the measurements return to a normal level, again high anomaly scores are produced (3). (based on [Bielefeld 2012a])

model, because the reference model of the common forecasting methods is not working properly in the case of collective anomalies. The goal is to prevent the sliding window of the forecasting method to be filled with abnormal values over the duration of a collective anomaly. To be able to still produce an anomaly score, at this point the alternative reference model is used.

ΘPADx stores all anomaly detection results in a database. This will be shown in detail in section 3.2.4. The data can be stored over a long time period, for example over weeks or month. This offers the opportunity to identify seasonal patterns even if they have a long period.

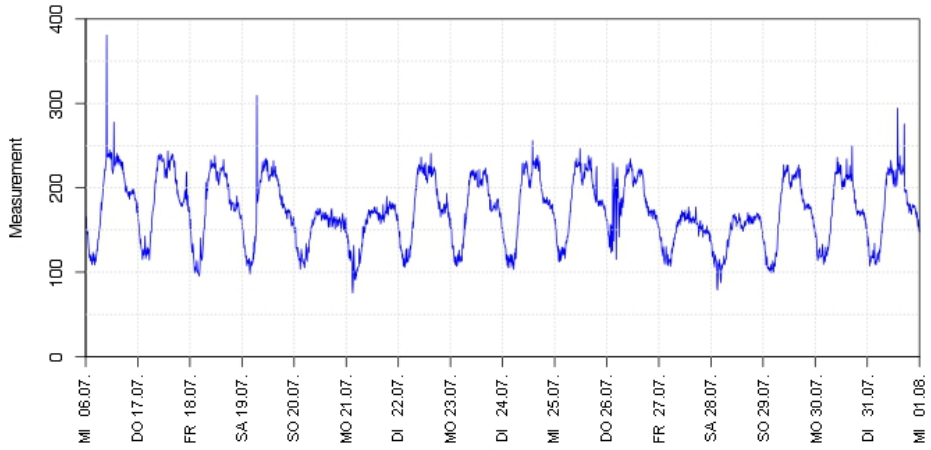

Figure 3.5. Overview of the perlapp over a duration of 16 days.

Figure 3.5 shows an example time series of the historic data of the Xing application *perlapp*. The shown measurements are total times in milliseconds. It can be seen that the workdays have a similar characteristic. In the morning hours the total time raises from around 100ms until it reaches circa 240ms. In the afternoon the values slowly go back down to 100ms. However, the weekend days show a different behavior. The measured values are way more constant during the day. They are around 160ms and only swing down to circa 100ms over the night. Due to this fact, a time point on Friday for example should not be used as reference for a time point on Saturday. Therefore, it would be appropriate to select a pattern with a weekly period. For each time series point a matching time series point, one or more weeks in the past, can be found with the help of this pattern. For this example, this is the base of the alternative pattern. Whenever a matching time series point in the past is found, a time series window is established at this point. This procedure is shown in Figure 3.6. A collective anomaly occurs, and the corresponding time points one and two weeks in the past are shown. A time series window of the same size as the original sliding window of the forecasting method is created. The found matching time point is the last value of this new time series window. A configurable aggregation strategy is then applied to the time series window and the resulting value is used as reference. Depending on how much values of the past are stored as historic data, matching time series points two weeks or even later may be found as well. These can be used if the first new created time series window also contains an anomaly, because if this is the case, it does not represent the normal system behavior.

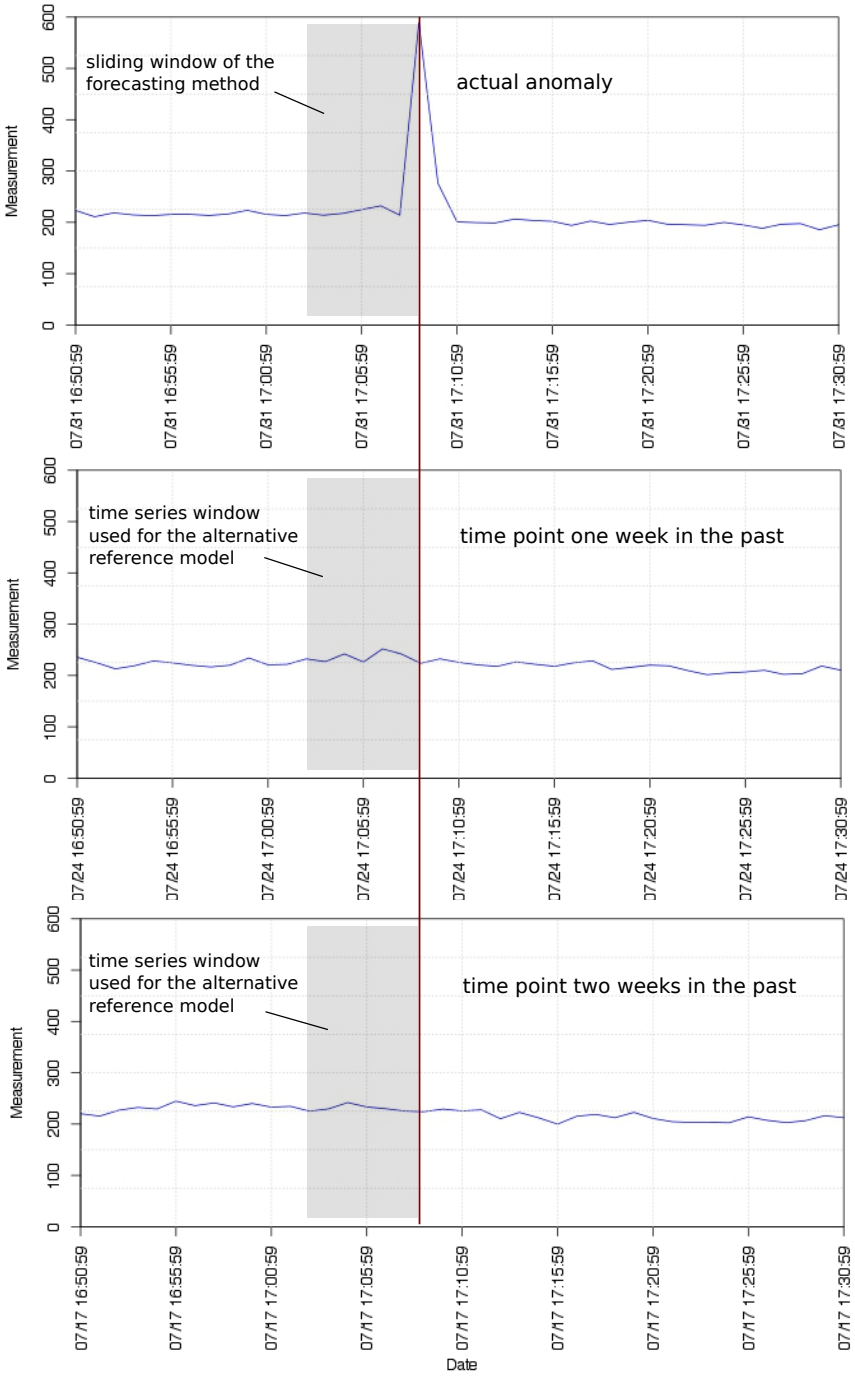

Figure 3.6. Matching time series points in the past are found and the corresponding sliding widows are created.

Forecasting (B2.1)

It is now known, how the pattern checking forecaster works. Since the forecasting step is responsible for the predictions, this is the step where the pattern checking forecaster is called, if appropriate. From the time series extraction activity (B1) discrete time series points are send to the forecasting step. This happens whenever the aggregation is completed, as described in Section 3.2.2. As soon as a new time series point arrives, a new prediction is calculated with the help of one of the presented forecasting methods (see Section 2.1.3) or, if a collective anomaly occurs, with the pattern checking forecaster. To decide whether to use the common forecasting methods or the pattern checking forecaster, the consecutive appearing anomalies are counted. Whenever this counter exceeds a configurable threshold, the forecasting step decides to use the pattern checking forecaster.

Since all used forecasting methods work on time series and not on single time series points, they need an opportunity to store the time series points. For this reason a sliding window of a configurable capacity is used. The sliding window stores the most recent time series points. Whenever the capacity of the sliding window is reached the oldest value is rejected. Since the value of the capacity is configurable, this is a further way to keep ΘPADx flexible. Besides this, it is also an opportunity to control the runtime of the forecasting methods, since this is basically the only input for them.

After the time series extraction activity a single discrete time series point is available for comparison. This also implies only the need of a single prediction, which implies the lead time $l = 1$ for the usage of all forecasting methods in the ΘPADx approach.

Unite Forecasting and Time Series Point (B2.2)

After the forecasting a single prediction that is based on the last received time series points is available. Since this is a prediction with lead time $l = 1$, its time index is one step in the future. This implies that we need to wait for the corresponding actual time series point. This simple task is realized in the unite forecasting and time series point step. It stores the forecast until the actual time series point is available and afterwards it unites both values in a record called *ForecastTimeSeriesPointPair* (FMP) of the form:

$$FMP = (x_{act}, f_{cor}),$$

Where x_{act} is the actual time series point and f_{cor} is the corresponding prediction. Whenever a FMP is completed, it is given to the anomaly score calculation step.

Anomaly Score Calculation (B2.3)

There are different approaches to calculate an anomaly score. As we will see in Chapter 4 the anomaly score calculation is implemented in a single filter. Due to the plugin structure

of Kieker, such filters can be exchanged easily. To use an alternative approach to calculate an anomaly score, simply a corresponding filter is needed. For the ΘPADx implementation the following way that was also introduced in Section 2.1.4, is used. In the past steps, the actual measurement was gained from raw data and is now available as a time series point. The reference model was gained in form of a one step ahead prediction. This means, that it is now possible to execute the comparison, which will result in the anomaly score. From the FMP x_{act} and f_{cor} are taken and used as input for the euclidean distance measurement as described in Section 2.1.4. This leads to the following calculation:

$$D_{euclid}(x_{act}, f_{cor}) = \sqrt{(x_{act} - f_{cor})^2}$$

Now the normalization, that was also presented in Section 2.1.4, is applied to finally gain the anomaly score A:

$$A(x_{act}, f_{cor}) = \left| \frac{D(x_{act}, f_{cor})}{x_{act} + f_{cor}} \right| \in [0, 1]$$

As soon as an anomaly score is calculated, it is given to the last step of this activity, the anomaly score interpretation.

Anomaly Score Interpretation (B2.4)

The anomaly score was already calculated in the previous step and forwarded to the final step of activity B2, the anomaly score interpretation. Here the score is compared to the anomaly threshold (see Section 2.1.4). If the anomaly score exceeds the anomaly threshold, it is classified as an anomaly, if not it is classified as normal. At this point the idea of anomaly detection pursued with ΘPADx is completed.

3.2.4 Send and Store Results (B3)

In this step the anomaly detection is already completed. The results can now be send to a connected facility and stored in a database. With the help of this activity, the results of the anomaly detection approach are sent to connected facilities. Several facilities are thinkable, for example a mail server that sends out an e-mail if an anomaly appears or a monitoring tool, that visualizes the anomaly detection results. For the case study environment at Xing, the results are send back to Logjam via ZeroMQ. The results sent contain the identifier, the anomaly score and a boolean value for the classification if anomaly or not. In Logjam the anomaly score is matched to the corresponding graph with the help of the identifier. The anomaly score is then shown right next to the graph. If the classifier indicates an anomaly, the score is rendered in red to gather the attention of an admin (see Figure 3.7).
After the data is sent, it is also stored in a data base. This data is then stated as historic data and can be used for example in the pattern checking forecaster (see Section 3.2.3). The stored data contains the actual time series point (discretizised), the identifier, the timestamp, the prediction or the pattern matched value, the anomaly score and a boolean

value for the classification if anomaly or not. The implementation details of the storage and send process are shown in the next chapter.

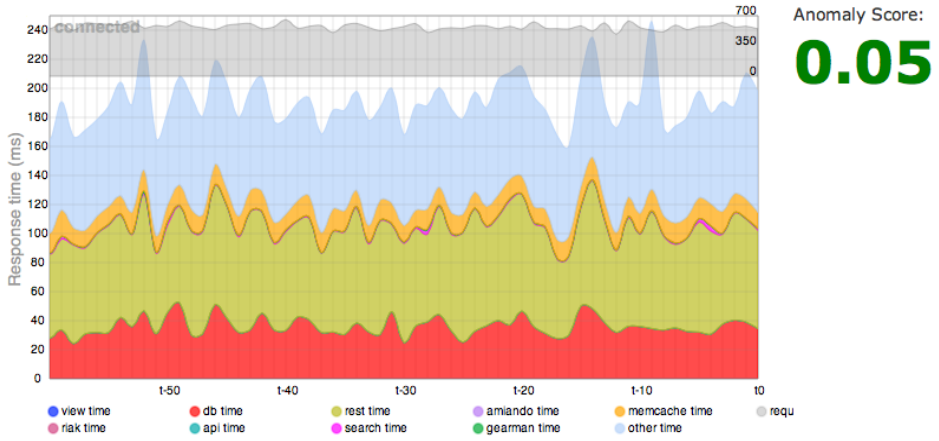

Figure 3.7. A graph of Logjam that shows the actual behavior of the selected application. The corresponding anomaly score is displayed live, right next to it.

Chapter 4

Implementation

This chapter will introduce the implementation of ΘPADx, which corresponds to the previous presented approach. This implementation is based on a refactored version of ΘPAD which fits to the pipes-and-filters plugin structure of Kieker. First, the goals of the implementation are presented. Then an overview of the most important components and the communication between them is given. After this, a more detailed look on all these components, including the adapter, the ΘPADx Kieker plugins and the storage is given.

4.1 Goals of the Implementation

The implementation of ΘPADx is based on the implementation of ΘPAD by Bielefeld [Bielefeld 2012a]. Therefore, it already fulfills several functional and non-functional requirements [Bielefeld 2012a, p. 47-49]. For the upcoming implementation, four additional functional requirements are necessary. These can be derived from the organizational goals of this thesis:

- **FR1: An improved anomaly detection approach**
  The ΘPADx approach, presented in Chapter 3, describes an improvement of the original ΘPAD approach. This improvement, that aims to fulfill G1 of the organizational goals, should now be included in the corresponding implementation.
- **FR2: Communication with the environment via ZeroMQ**
  This requirement corresponds to G3.1 and can, as will be shown later on in this chapter, be fulfilled by the development of an adjusted adapter.
- **FR3: Support for multiple applications per ΘPADx instance**
  A single instance of ΘPADx should be able to process the data of several applications as required in G3.2. This way a better system coverage is possible in contrast to process only the data of a single application as possible with ΘPAD. Also a more detailed look on certain parts of the software system can be achieved with the help of this feature.
- **FR4: Support for multiple metrics per application**
  This last additional requirement is derived from G3.3. ΘPAD is only capable to process a single metric of the selected application. The implementation of ΘPADx should be able to process several metrics per application.

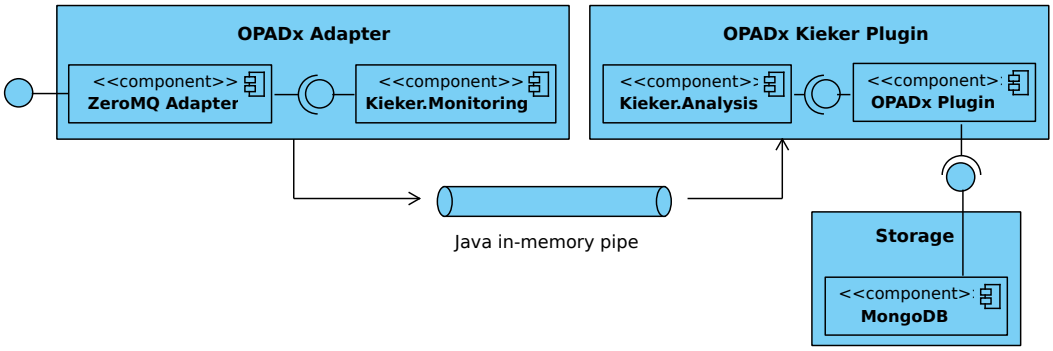

Figure 4.1. Overview of the ΘPADx architecture.

4.2 Implementation Overview

The ΘPADx implementation consists of three main components: the adapter, the ΘPADx Kieker plugins, and the storage. Figure 4.1 gives an overview of the components and how they work together to form the ΘPADx implementation. To make use of Kieker and ΘPADx simply a Kieker.jar file, that also contains the ΘPADx Kieker plugins, must be available within the corresponding classpath of the adapter.

The adapter is able to receive data from the environment. In this case the data is sent via ZeroMQ. Within the adapter the gathered data is transformed to a kind of so-called Monitoring Records. These records are provided by the Kieker Monitoring component and can be further processed by Kieker plugins. More details about the records are given in Section 4.3. In this implementation an in-memory pipe, provided by Kieker, is used to pass the records from the Monitoring component to the ΘPADx plugins. Kieker also provides further methods for this purpose [Kieker Project 2013].

The ΘPADx Kieker plugins execute the actual anomaly detection. The plugins are connected according to the pipes-and-filters pattern: it consists of several filters that are connected via pipes. The filters correspond to the activities presented in Chapter 3.

In the last filter the results of the anomaly detection are stored to a MongoDB instance. Therefore, this storage contains the results of every anomaly detection done. This includes not only the anomaly score, but also additional data. A detailed look on the stored data is given in Section 4.5.

4.3 Adapter

This section introduces the advantages of the adapter concept in general. After that, the implementation of the adapter used at the case study environment is presented.

4.3.1 The Adapter Concept

In programming the adapter concept can be found for example in the field of design patterns [Gamma et al. 1995]. The so-called *adapter pattern* has the intention to "let classes work together that couldn't otherwise because of incompatible interfaces" [Gamma et al. 1995]. Here the concept is used to let the data of a software system work together with Kieker and the ΘPADx Kieker plugins. In this case, data can be send with a common communication technology and is transformed into Kieker Monitoring Records that are processed by the ΘPADx plugins.

The adapter concept leads to a higher flexibility for the usage of the implementation. If an alternative communication technology is used, simply a corresponding adapter needs to be developed. This can be achieved without touching the core implementation of the anomaly detection approach itself. Overall, this leads to a clear separation between the logic of the anomaly detection approach and the adapter, that is responsible to collect the data from the environment.

4.3.2 Adapter for the Case Study Environment

As seen in Section 2.4.3, the monitoring data of Xing is communicated via ZeroMQ. The adapter of the ΘPADx implementation is used to gather data from a software system of interest. For the case study environment at Xing an adapter was developed that is able to receive data via ZeroMQ. All the available monitoring data, especially the performance data, can therefore easily be communicated to ΘPADx. With the help of this adapter FR2 can be accomplished.

ZeroMQ offers built-in support for several communication patterns (see Section 2.4.3). One of these patterns is the Publish-Subscribe pattern. A subscriber can register itself to a publisher, if it is interested in its data. The publisher sends out data to all registered subscribers. The Xing monitoring architecture already contains a publisher that sends out all available monitoring data. If only a subset of this data is needed, a subscriber can define so-called filters to receive only the data that matches to this filters.

With this prerequisites the adapter can be implemented as subscriber with certain filters to be able to receive the desired performance data. The following listing shows a part of an example message received by the adapter via ZeroMQ.

```
1 perlapp-production,all_pages
2 {
3         "count":880.0,
4         "other_time":64415.4899731293,
5         "other_time_sq":16864373.3219688,
6         "memcache_time":22147.961,
7         "memcache_time_sq":2589201.320141,
8         "rest_time":61713.078,
9         "rest_time_sq":31111680.995692,
10        "db_time":42402.8990268707,
11        "db_time_sq":12164506.6471881,
12        "total_time":191525.086,
13        ...
14 }
```

Figure 4.2. A part of a JSON message received by the adapter.

The filter this message corresponds to is mentioned in the first line. The incoming messages are encoded as JSON strings. The values needed can therefore be extracted easily with the help of a Java JSON library[1].

As mentioned, the ΘPADx implementation supports the anomaly detection for several applications with only one instance. Therefore, a unique identifier per application is needed, to be able to separate the incoming data. For this implementation the name of the application is used as identifier. Optionally, the name of an application in combination with a desired metric can form an identifier. The performance data of the here considered applications and metrics can be interpreted as double value. This leads to the implementation of a special kind of Monitoring Records, the so-called `NamedDoubleRecords`. This kind of record extends the `AbstractMonitoringRecord` class and implements the `IMonitoringRecord` interface as determined by the Kieker Framework. Overall, these records wrap an extracted value and the name of the application the value is assigned to. These records are then deserialized and a Java in-memory pipe is used by the Kieker Monitoring component to forward them to the ΘPADx plugins. In detail, a so-called `PipeWriter` puts the messages into the pipe. This type of communication can be activated within the `kieker.monitoring.properties` file, that is responsible for the properties of the Kieker Monitoring component. The corresponding entry to the file is shown in Listing 4.3.

The Kieker Analysis component, in which the Kieker plugins are located and therefore also

[1]For this implementation, the following JSON library was used: http://www.json.org/java/

the ΘPADx plugins, is then able to read from this pipe. In Section 4.4.2 the configuration of the reader for the "opad-pipe" is presented.

```
1 kieker.monitoring.writer=kieker.monitoring.writer.namedRecordPipe.PipeWriter
2 kieker.monitoring.writer.namedRecordPipe.PipeWriter.pipeName=opad-pipe
```

Figure 4.3. Configuration of the writer of the Kieker Monitoring component. The `PipeWriter` is selected and the name of the pipe is set to "opad-pipe".

4.4 ΘPADx Kieker Plugins

In this section the implementation of the ΘPADx Kieker plugins is presented in detail. The plugins contain the core implementation of the approach presented in Chapter 3. First the prerequisites for the implementation of a Kieker plugin are shown. After that, the structure of the ΘPADx plugins and their most important features are described.

4.4.1 Kieker Plugin Prerequisites

As mentioned, the core of the ΘPADx approach is implemented as Kieker plugins. Every Kieker plugin is located in the Kieker Analysis component. Kieker plugins can be categorized in readers, filters, and repositories and can be connected according to the pipes-and-filters architecture. To develop for example a custom filter, the corresponding abstract class `AbstractFilterPlugin` must be extended as determined by Kieker. A filter has a name, output ports, and several configuration properties. All these values can be defined comfortably with annotations. Listing 4.4 shows a part of the `ExtractionFilter`, that is used within ΘPADx. The extension from the `AbstractFilterPlugin` can be seen, as well as the used annotations.

The following steps are necessary to start an analysis with developed plugins. First an instance of the `AnalysisController` class, provided by Kieker, is created. Then the readers, filters and repositories are created. These are now configured with the help of the properties, defined through the `@Plugin` annotation. If no configuration is made, default values are chosen instead. The created and configured readers, filters and repositories are then connected with each other. In a final step the analysis is started.

4.4.2 ΘPADx Plugins - Structure and Features

The previous section showed, that Kieker plugins are readers, filters or repositories, that can be combined to a pipes-and-filters architecture. The ΘPADx Kieker plugins also follow this structure. For the implementation of ΘPADx a reader and several filters were used. In

```
@Plugin(
        name = "Extraction_Filter",
        outputPorts = {
                @OutputPort(name = ExtractionFilter.OUTPUT_PORT_NAME_VALUE,
                        eventTypes = { NamedDoubleTimeSeriesPoint.class }) },
        configuration = {
                @Property(name = ExtractionFilter.CONFIG_PROPERTY_NAME_TIMEUNIT,
                        defaultValue = "NANOSECONDS")
        })

public class ExtractionFilter extends AbstractFilterPlugin {
...
```

Figure 4.4. A part of the `ExtractionFilter` of the ΘPADx implementation. The class extends the `AbstractFilterPlugin` and defines a name, an output port and a configuration property with the help of annotations.

the following sections, the architecture of the implementation is presented and the most important details of the reader and the different filters are shown.

Data Flow

In Section 2.5, the pipes-and-filters architecture of the refactored ΘPAD implementation was briefly introduced. It can be seen in Figure 4.6 that the architecture of the ΘPADx plugins is quite similar to this. Hence, the goals of the implementation are not achieved by changes to the overall structure of the plugins. Instead they are achieved by changes to the already available filters themselves and the usage of an additional forecaster.

The pipes-and-filters architecture of the ΘPADx plugins starts with a reader. This `PipeReader` is necessary to read from the in-memory pipe, that was also used in the Adapter component. The reader is configured as shown in Listing 4.5. With the help of the name of the corresponding pipe, in this case the "opad-pipe", the connection between the adapter and the ΘPADx plugins is achieved. The data is then forwarded to the first filter. Every

```
final Configuration pipeConfig = new Configuration();
pipeConfig.setProperty(PipeReader.CONFIG_PROPERTY_NAME_PIPENAME, "opad-pipe");
final PipeReader reader = new PipeReader(pipeConfig, controller);
```

Figure 4.5. Configuration of the `PipeReader`. With the help of the pipe's name, the data sent by the adapter can be assigned.

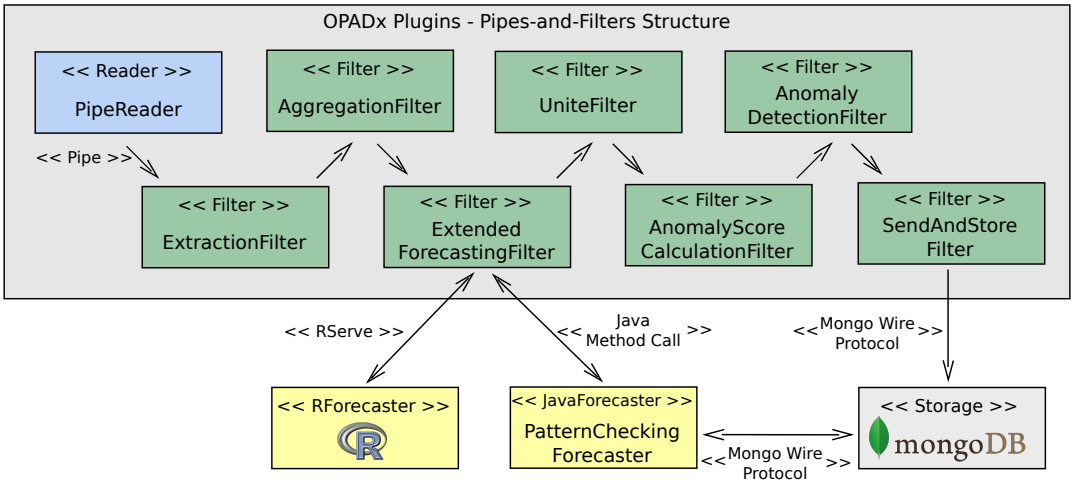

Figure 4.6. The data flow of the ΘPADx plugins. The filters manipulate the data and the data is forwarded with the help of pipes. The `ExtendedForcastingFilter` uses *R* via *Rserve* for the forecastings or optionally the pattern checking forecaster, implemented in Java.

filter corresponds to an activity or a step presented in the approach chapter and offers the described functionality. This way the data is manipulated by each filter and forwarded with the help of the pipes.

TSLib Component

While the data is forwarded from filter to filter, it is manipulated. For example, if the data enters the extraction filter, it is transformed from a `NamedDoubleRecord` to a `NamedDoubleTimeSeriesPoint`. Another example is the forecasting filter. Whenever this filter is passed, the data is enriched by the forecasting result. For all these purposes special data structures are necessary. Therefore, Bielefeld bundled basic time series related data structures in a library called TSLib, that in the meantime is maintained as part of the Kieker framework. The provided classes can be seen in Figure 4.7. This way, all the needed data structures can be derived by the basic data structures of the TSLib.

Beside this, the TSLib contains interfaces for the used forecasting methods and provides support for a connection to R. This way, computation-heavy forecasting methods can be calculated in a separate R instance. A detailed look on this connection is given in Section 4.4.2.

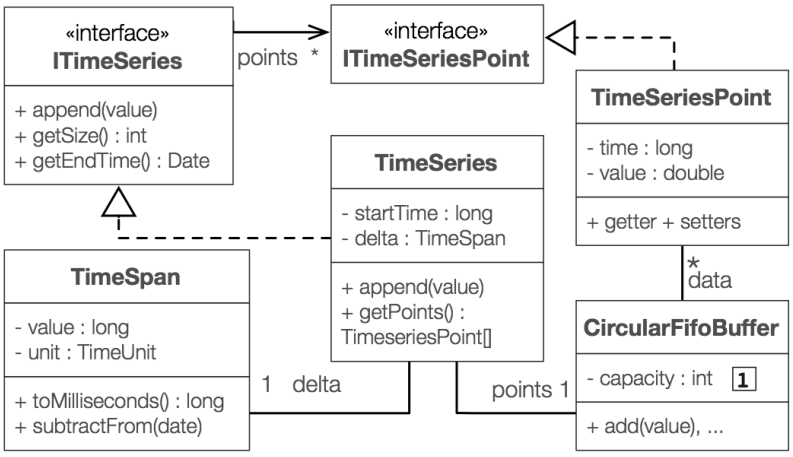

Figure 4.7. Classes provided by the TSLib, depicted as UML class diagram. (Image by [Bielefeld 2012a])

Separation of Applications and Metrics

It was already mentioned, that the data sent to the ΘPADx plugins provides an identifier. The filters use this identifier to separate the data of the different applications and, if needed, of the different metrics per application. This behavior of the implementation was also demanded in FR3 and FR4

To fulfill these requirements some of the filters need an additional way to store data, which depends on the actual processed identifier. For example, the `AggregationFilter` needs to store the values, that were already collected within a aggregation timespan. Whenever two consecutive incoming values do not belong to the same identifier, these values are not allowed to be mixed. Another example is the `ExtendedForecastingFilter`, where the sliding windows of the different identifiers are not allowed to be mixed, because otherwise the forecasting result would become useless.

For this purpose, the filters of ΘPADx store the above mentioned data in a `ConcurrentHashMap`. Since the identifiers are unique, they can be used directly as keys to store or request a value to or from the `ConcurrentHashMap`. In Listing 4.8 a part of the `ExtendedForecastingFilter` is shown as example for the implementation of the separation. The function *inputEvent()* is called whenever a value enters the filter. First, it is checked if data already exist for the actual identifier (line 4). If data already exist, the input can be processed (line 5), if not, a new entry must be created within the `ConcurrentHashMap` named *applicationForecastingWindow* (lines 7-11).

```
1 @InputPort(eventTypes = { NamedDoubleTimeSeriesPoint.class },
2         name = VariateForecastingFilter.INPUT_PORT_NAME_TSPOINT)
3 public void inputEvent(final NamedDoubleTimeSeriesPoint input) {
4         if (this.checkInitialization(input.getIdentifier())) {
5                 this.processInput(input, input.getTime(), input.getIdentifier());
6         } else {
7                 // Initialization of a sliding window for a new application
8                 this.applicationForecastingWindow.put(input.getIdentifier(),
9                 new TimeSeries<Double>(System.currentTimeMillis(), this.deltat,
10                        this.tunit, this.timeSeriesWindowCapacity));
11                this.processInput(input, input.getTime(), input.getIdentifier());
12        }
13 }
14
15 private boolean checkInitialization(final String name) {
16        return this.applicationForecastingWindow.containsKey(name);
17 }
```

Figure 4.8. A part of the `ExtendedForecastingFilter`, which shows how the data is separated with the help of identifiers and `ConcurrentHashMaps`.

Extended Forecasting Filter and Pattern Checking

The main extensions of ΘPADx are located in the `ExtendedForecastingFilter` and the pattern checking forecasting method. These extensions aim to improve the anomaly detection results of the approach and therefore address FR1. In Chapter 3 the pattern checking was presented. This forecasting method only takes place if a collective anomaly occurs, since this is the case in which the ΘPAD approach produces an amount of unnecessary false positives. Therefore, the `ExtendedForecastingFilter` was created. This Filter is able to use the pattern checking forecasting method and can also decide when this method is used over the other forecasting methods. In the following the working method of the `ExtendedForecastingFilter` and the corresponding forecasting methods is shown. The activity diagram in Figure 4.9 gives a first overview of the processes in the filter.

When a value enters the `ExtendedForecastingFilter`, first the actual sliding window for the corresponding identifier is loaded, as shown in the previous section. To decide if a collective anomaly is present or not, a counter that stores the amount of consecutive occurring anomalies is used. This information is also stored separately for every identifier within the `ExtendedForecastingFilter`.

Based on a configurable threshold and the counter of consecutive anomalies, the `Extended-ForecastingFilter` chooses between the configured forecasting method and the pattern

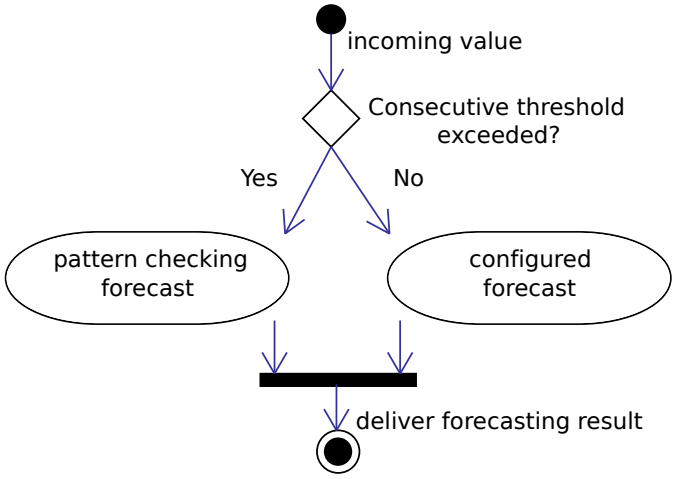

Figure 4.9. This activity diagram depicts the decision between the pattern checking forecaster and the configured forecasting method in dependence on the amount of consecutive anomalies.

checking forecasting method (see Section 3.2.3).

The pattern checking forecaster is implemented in Java, as a class called `PatternCheckingForecaster`. As mentioned, the TSLib provides interfaces for the usage of forecasters. The implementation of the `PatternCheckingForecaster` therefore extends the `AbstractForecaster<T>` class, that is provided by the TSLib. This enables the `ExtendedForecastingFilter` to make use of this forecaster. The forecasting method of the `PatternCheckingForecaster` is called with parameters that are used to determine the pattern. These parameters can be set in the configuration of the `ExtendedForecastingFilter`. With the help of the parameters, a query to the storage is sent, that returns a time series window that corresponds to the configured pattern, as was also described in Section 3.2.3. If this is successful, the result of the `PatternCheckingForecaster` is used.

If no consecutive anomaly is detected, the configured forecasting method is used. These forecasting methods are already implemented in the forecasting package of *R*. For the connection to *R*, Rserve[2] is used. The library JRClient[3] Wraps this connection and additionally deals with the transformation of Java data structures to *R* data structures. With JRClient it is then possible to directly call *R* methods from a Java application. To simplify the development of additional forecasters, Bielefeld created an interface for the communication of forecasters with *R* and added it to the TSLib. A forecaster that implements this interface, called `IForecaster`, is able to use the corresponding method of the *R* forecasting package

[2] http://www.rforge.net/Rserve/
[3] http://stats.math.uni-augsburg.de/rserve/dist/JRclient/JavaDoc/

and can also be used from the `ExtendedForecastingFilter`.

4.5 Storage and Send

For every finished anomaly detection process, the results are stored in a MongoDB instance (see Section 2.4.2). The stored data has two key tasks within the ΘPADx implementation. First of all, the stored results of the anomaly detection can be used for example to do offline analysis or create visualizations. However, in contrast to ΘPAD, the storage of ΘPADx is also used for forecasts with the pattern checking forecaster, as shown in 4.4.2.

The storage of only a small set of data per anomaly detection result is necessary to be able to accomplish the mentioned two tasks:

- **Identifier**
  The identifier is stored to be able to assign the rest of the result to an application and optionally a metric.

- **Value**
  In this part of the result, the value of the incoming measurements after the aggregation is stored. This value is needed if the time series of original values should be recreated at a later stage, e.g. for an offline analysis.

- **Timestamp**
  Here the timestamp that is generated right after the aggregation, is stored. With this timestamp it is possible to order the values chronological. On this way, it is possible to recreate the original time series from the results data.

- **Forecast**
  In addition to the values also the corresponding forecasts are stored. This allows for example to compare the values with the forecasts in an offline analysis.

- **Score**
  In addition to the values and the forecasts, also the calculated score is stored. This is especially interesting to compare different configurations of ΘPADx with each other.

- **Anomaly**
  This part of the result contains a boolean value, that is set to true if the stored result was interpreted as anomaly and false if not. This value is used for the database queries of the pattern checking forecaster and the interpretation and graphical visualization of the result in logjam (red font color if true, black if false).

The following listing shows example MongoDB entries, that consist of the mentioned parts:

```
...
{
"_id" : ObjectId("5226f9500364de301c55f7ab"),
"identifier" : "companies-production,corporatepages",
"value" : 133.41654450094734,
"timestamp" : NumberLong("1378285901848"),
"forecast" : 133.41654450094734,
"score" : 0.02113034255302418,
"anomaly" : false
}

{
"_id" : ObjectId("5226f98b0364de301c55f7b9"),
"identifier" : "messages-production,all_pages",
"value" : 202.70827268562502,
"timestamp" : NumberLong("1378285962046"),
"forecast" : 222.48932688551184,
"score" : 0.04652202698189832,
"anomaly" : false
}
...
```

Figure 4.10. The listing shows sample results of the ΘPADx anomaly detection, as they are stored in a MongoDB instance. Based on the identifiers it can be seen that the two entries belong to two different applications.

For the case study environment at Xing the results are sent back to Logjam for monitoring purposes. Therefore, the identifier, the score and the anomaly flag of the result are combined to a `JSONObject` and sent via ZeroMQ as shown in Listing 4.11. In Logjam the anomaly detection result can be assigned to the corresponding application with the help of the identifier.

```
final JSONObject obj = new JSONObject();
final String key = detectionData.getIdentifier();
obj.put("score", detectionData.getScore());
obj.put("anomaly", true);
this.publisher.send(key, ZMQ.SNDMORE);
this.publisher.send(obj.toString(), 0);
```

Figure 4.11. The implementation of the send process if an anomaly was detected (anomaly flag set to true).

Chapter 5

Evaluation

In this chapter, the evaluation of the introduced anomaly detection approach and the corresponding implementation is presented. This contains the definition of conceptual goals and metrics according to GQM. Additionally, the set-up and execution of a back-to-back test and a long-term test in the evaluation environment is shown. Finally, an analysis of the received anomaly detection results with respect to the defined goals is made.

5.1 GQM Evaluation Plan

As mentioned in the introduction this thesis follows the GQM approach. In Chapter 1 the organisational goals (G1 – G4) of this thesis were defined. As the organisational goals are known, the evaluation aims to find conceptual goals (CG). These are defined to split up G1 – G4 into sub-goals that are of finer granularity. According to GQM, questions are found for every conceptual goal to associate it with a set of data. With the context given by the organisational and conceptual goals and the questions, fitting metrics will be derived. The evaluation chapter of Bielefeld [Bielefeld 2012a] was also structured according to the GQM approach. He already stated that the following conceptual goals have to be fulfilled by the evaluation of an online anomaly detection approach and the corresponding implementation:

- **CG1**: Determine a configuration of ΘPADx (activated / deactivated extension) for the case study environment
- **CG2**: Evaluate the practicality of ΘPADx
- **CG3**: Carry out an anomaly causality research

The ΘPADx approach is based on ΘPAD, but as shown in the previous two chapters, the approach and also the corresponding implementation are extended. Given this fact, the mentioned conceptional goals will be a part of the evaluation process to show that ΘPADx is still usable in a production environment. Since without the activated extension ΘPADx is basically ΘPAD for multiple applications a back-to-back test will be performed on the data gathered by Bielefeld [Bielefeld 2012a]. This will ensure that ΘPADx still works as intended for the single application use case. One goal defined in Chapter 1, is the improvement of the anomaly detection results. As mentioned in Chapter 4, it is possible to use or not to

use the extended anomaly detection approach of ΘPADx because of the flexible Kieker pipes-and-filters plugin structure. Due to the extension a comparative evaluation between ΘPADx with and without activated extension can be performed. For the final assessment of the two approaches a comparison on an actual collected data set and the data set used by Bielefeld is made. Therefore, this additional conceptual goal is defined:

▻ **CG4**: Assess the extensions made to the anomaly detection process

According to the GQM paradigm questions are assigned to the conceptual goals. To decide whether these goals are fulfilled or not, measurable metrics must be defined as well.

5.2 Conceptional Goals

In this section conceptual goals are defined in detail. They all refer to an object, that is assessed, and cover several corresponding questions. For this evaluation the objects are the ΘPADx approach, the pair of ΘPAD and ΘPADx and finally the case study environment. To be able to assess the goals, every question within a goal needs corresponding metrics. The metrics used here are introduced by Bielefeld [Bielefeld 2012a], because of the close relation between ΘPAD and ΘPADx these metrics are also fitting to this evaluation process. These metrics are introduced after the definition of the conceptual goals.

CG1: Determine a configuration of ΘPADx (activated / deactivated extension) for the case study environment

The process of anomaly detection used by ΘPADx relies on several parameters (see Chapter 3). If this goal is fulfilled, a combination of the parameters is found that allows to run ΘPADx in the context of the case study environment.

Conceptional Goal 1 (CG1)		
	Purpose	Find a good
	Issue	configuration setup of
	Object	ΘPADx
	Viewpoint	for the case study environment
Question	Q1.1	Which algorithms to use?
	Q1.2	Which threshold detects good?
	Q1.3	Which aggregation span is good?
	Q1.4	Which sliding window length is good?
Metrics	M3	True positive rate (TPR)
	M4	False positive rate (FPR)
	M8	F-measure
Question	Q1.5	Which performance attributes matter?
Metrics	M11	Anomaly and attribute correlation
	M12	Cause of an anomaly
	M13	Anomaly duration
	M14	Anomaly classes

CG2: Evaluate the Practicality of ΘPADx

ΘPADx is designed to perform online anomaly detection. This goal is created to see if the approach is able to achieve its main use case in a general manner.

Conceptional Goal 2 (CG2)		
	Purpose	Assess
	Issue	the practicability of
	Object	extended performance anomaly detection
	Viewpoint	on an online, automatic basis
Question	Q2.1	Is the ΘPADx server stable?
Metrics	M1	Server uptime
	M2	CPU utilization
Questions	Q2.2	How precise is the detection?
	Q2.3	How accurate is the detection?
	Q2.4	How good is the overall quality of the detection results?
Metrics	M6	Precision
	M7	Accuracy
	M8	F-Measure

CG3: Anomaly causality research

ΘPADx is able to collect a huge amount of data from large-scale software systems. To be able to identify occurred anomalies it is necessary to determine which factors cause anomalies in such systems.

Conceptional Goal 3 (CG3)		
	Purpose	Learn which
	Issue	anomalies occur within
	Object	software systems
	Viewpoint	of large-scale architectures
Question	Q3.1	Which types of real anomalies occurred?
Metrics	M12	Cause of an anomaly
	M13	Anomaly duration
	M14	Anomaly classes

CG4: Assess the extensions made to the anomaly detection process

ΘPADx brings several extensions to the ΘPAD approach. While the opportunity of processing analysis for multiple applications and metrics is more of a conceptional feature, ΘPADx also extends the anomaly detection approach. Thus, this goal is defined to assess these two kinds of extensions.

Conceptional Goal 4 (CG4)		
	Purpose	Assess
	Issue	the differences of
	Object	the ΘPAD and ΘPADx approach
	Viewpoint	for the use in practice
Question	Q4.1	Are the additional conceptual features useful?
Metrics	M15	Usability of the ΘPADx approach
Question	Q4.2	How differs the quality of the detection results?
Metrics	M5	Number of actual anomalies
	M9	Number of true negatives
	M10	Number of false positives

5.3 Metrics

To be able to evaluate the results that correspond to the conceptional goals, measurable values are required. The following metrics are defined to give the opportunity to compare and assess the results of the evaluation. In the following, the metrics are divided into two groups, objective metrics and subjective metrics, as defined in Section 1.3.1.

5.3.1 Objective Metrics

- **M1 - Server Uptime**
  The time is measured, in which the processes, that are part of the ΘPADx implementation, were running and available.

- **M2 - CPU Utilization**
  The CPU utilzation is measured with the Unix `top` command. The utilization is observed during the aggregation span, when the amount of calculations is low, and during the time with an higher amount of calculations directly after the aggregation span is completed. Therefore, a range is used for the CPU utilization results.

- **M3 - True Positive Rate**
  The true positive rate was already defined in Section 2.2.1.

- **M4 - False Positive Rate**
  The false positive rate was already defined in Section 2.2.1.

- **M5 - Number of Actual Anomalies**
  The number of actual identified anomalies during the evaluation interview.

- **M6 - Precision**
  The precision was already defined in Section 2.2.1.

- **M7 - Accuracy**
  The accuracy was already defined in Section 2.2.1.

- **M8 - F-Measure**
  The F-measure was already defined in Section 2.2.1.

- **M9 - Number of True Negatives**
  The amount of true negatives, interpreted by the ΘPADx approach.

- **M10 - Number of False Positives**
  The amount of false positives, interpreted by the ΘPADx approach.

5.3.2 Subjective Metrics

- **M11 - Anomaly and Attribute Correlation**
  Attributes, that can be used to find correlations between anomalies.
- **M12 - Cause of an Anomaly**
  A factor that can be determined as cause of an anomaly.
- **M13 - Anomaly Duration**
  The duration of an anomaly in a timely manner. The duration of the identified anomalies is determined with the help of graphs by system experts, for example in the evaluation interview. Because of this, the duration is here classified as subjective.
- **M14 - Anomaly Classes**
  The anomalies are clustered in different often appearing classes during the evaluation process.
- **M15 - ΘPADx Usability**
  The usability of the ΘPADx implementation, that is mainly determined by interviewing the administrators using the implementation and the interpretation of the resulting visualizations.

5.4 Evaluation Process

The evaluation process consists of several steps. First a back-to-back test on the data gathered by Bielefeld will be done. After this, ΘPADx is set up and started to collect actual data of the case study environment over a given time. For the evaluation, this enables the opportunity to run offline analysis on the same data set as often as needed. Besides this, an evaluation interview is held to identify possible anomalies that occurred within the time period of the data collection. With the results of the evaluation interview the quality of the anomaly detection can be determined. With several replays of the analysis, an optimized configuration will be found that fits to the case study environment. This is achieved by a comparison of the anomaly detection results of all runs with the help of the detection comparison metrics (see Section 2.2.1). The quality of the anomaly detection of ΘPADx, under the best configuration found, can then finally be compared to the results of ΘPAD.

5.4.1 Back-to-Back Test

Within this section a back-to-back test of ΘPADx and ΘPAD, on the data that was gathered by Bielefeld, is presented. First ΘPADx will run on this data set with the best found configuration by Bielefeld. Since ΘPADx is based on ΘPAD, this should result in equal results as found by Bielefeld. After this, ΘPADx is configured to run with the active extension. The different results can then be compared to see if the extension is influencing

Table 5.1. The manually identified anomalies of the data set gathered by Bielefeld.

Back-to-Back Test Anomaly Intervals			
Start	End	Duration	Attribute
19.12. 14:05	19.12. 14:46	00:41	count
21.12. 13:47	21.12. 15:09	01:22	db_time
21.12. 23:08	21.12. 23:20	00:12	count
22.12. 09:33	22.12. 09:35	00:02	db_time
23.12. 13:04	23.12. 13:06	00:02	view_time
27.12. 09:45	27.12. 10:15	00:30	count
29.12. 15:08	29.12. 15:31	00:23	api_time
30.12. 01:48	30.12. 02:09	00:21	db+api_time

Table 5.2. The best found parameters found by Bielefeld for this data set.

Back-to-Back Test Parameters	
Forecasting algorithm	Mean
Aggregation time	1 minute
Forecasting time window	1 hour
Threshold	0.23

the anomaly detection results. As mentioned, Bielefeld gathered the data of a single application. Therefore, this back-to-back test covers the use case of a single application anomaly detection.

Data Set and Configuration

The data by Bielefeld [Bielefeld 2012a] was collected over 12 days from 18.12.2011 to 30.12.2011. There were 8 anomalies detected manually in the process of an evaluation interview. The intervals in which the anomalies appear are presented in Table 5.1. As can be seen from the duration of the anomaly intervals, the major part were collective anomalies. These were found to be the cause of additional false positives as described in Section 3.2.3. For a more detailed description of the anomalies the evaluation chapter of Bielefeld [Bielefeld 2012a] can be consulted.
The configuration that is presented in Table 5.2 was found to be the best on this data set. The chosen forecaster is the mean forecaster with a sliding window of 1 hour. The incoming raw data was aggregated over one minute, while the anomaly threshold was set to 0.23. In an additional anomaly detection run now also the pattern checking is activated.

Observations

With the configuration as shown in Table 5.2 Bielefeld got the following results. A precision of $PREC = 0.14$ while the accuracy reached a value of $ACC = 0.97$. The anomaly detection with ΘPADx without extension provides the exact same results for this use case (B1). Additionally, an F-measure of 0.15 was calculated.

During the process of the back-to-back test a problem with the anomaly intervals was found. The timestamps of the anomalies in the data did not correspond to the timestamps of the manually identified anomalies. Since the timestamps were shifted by exactly one hour, the best possible explanation for this is a problem with the time zones in the process of generating the timestamps. Therefore, the back-to-back test was repeated with corrected timestamps (B2). Finally, ΘPADx was configured to run with activated pattern checking on the data set with corrected anomaly intervals (B3). Therefore a pattern parameter of the anomaly detection was set to 24 hours. The results of all runs are discussed in detail in Section 5.5, together with the results of the evaluation on the actual data collection.

5.4.2 Actual Data Collection

An implementation of the ΘPADx approach is presented in Chapter 4. While and after the process of development, several testing phases were carried out to ensure the stability of the implementation. However, the data collection exceeded the runtime of all previous stability tests. Therefore, it was used to finally confirm the stability of the implementation. While collecting the data, ΘPADx was running with a configuration that is similar to the configuration that reached the best results in the evaluation of Bielefeld [Bielefeld 2012a]. This configuration is executing the anomaly detection constantly and additionally ensures that ΘPADx is collecting the incoming raw data. In this way the collection phase emulates the usage of ΘPADx in production environment over its runtime. The most important parameters chosen for the data collection are shown in Table 5.3. However, the main goal of the data collection was to gather the raw data of incoming values from the Xing environment that solves as case study. With this sample set of data, it is possible to rerun the anomaly detection in offline mode as often as needed. This is used during the evaluation to compare the results of the analysis with different configurations, activated or deactivated extensions and ΘPAD. Since ΘPADx is able to process data of several applications, it was set up to gather raw data from 14 different applications. For some of the applications different pages are considered as well. An exact list of all selected application and page combinations is shown in Table 5.4.

For the data collection ΘPADx was deployed on a server within the Xing cluster (Debian, CPU, RAM). The server runs Java 1.6, which is the programming language ΘPADx is written in. As mentioned in Chapter 2.4.3 ΘPADx uses ZeroMQ to connect to the

Table 5.3. This table shows the most important parameters of ΘPADx used for the data collection.

ΘPADx Data Collection Parameters	
Time	
Start date	15.07.2013
End date	02.08.2013
ΘPADx Parameters	
Forecasting algorithm	JavaMean
Aggregation time	1 minute
Forecasting time window	1 hour
Threshold	0.23
Data	
Raw data storage	active
Analysis result storage	active
Server	
OS	Debian
CPU	Changing (virtually)
Memory	Changing (virtually)

logging queue. Via this logging queue the raw data of the Xing environment is collected. Therefore ZeroMQ is installed on the server. To store the raw data, a database connection is established in the adapter (see Chapter 4). When a raw measurement arrives in the adapter, a copy is stored to a separate collection of the MongoDB that is also used to store the anomaly detection results. Due to this location, no aggregation or other manipulation of the incoming raw data happened until this point. An installation of MongoDB on the server was therefore also required. The following Table 5.3 sums up the most important parameters of the ΘPADx instance that runs during the data collection phase.

5.4.3 Evaluation Interview

After the data collection phase an evaluation interview was scheduled for a manual analysis of the data that was collected. To gain the best possible results from the interview, it was held with Dr. Stefan Kaes, who is the head of the architecture team at Xing and worked with the underlying system for many years. Therefore, he has profound knowledge of the case study environment, its behavior and the data it produces.

The corresponding data from Monday the 15.07.2013 to Friday the 02.08.2013 was revisited in logjam. For a more detailed look on some intervals, plots with R were used.

Table 5.4. This table shows all application and page combinations that were selected during the data collection phase to gather data from.

Applications and Pages	
Application	Pages
companies-production	corporatepages, rest
events-production	events, rest
ipad-production	ipad
jobs-production	jobs, rest
messages-production	all_pages
perlapp-production	all_pages
perlrest-production	all_pages
perlyaml-production	all_pages
profiles-production	profiles
publicsearch-production	all_pages
recommenders-production	all_pages
riaktivities-production	rest
triton-production	webservice
xws-production	webservice, xws

The interview focused the data of the perlapp application under the all_pages selector. Despite the fact that Xing's main application got split up, the perlapp application is still the most important application and, therefore, the most important indicator for the health of the Xing platform. The amount of time for the interview and the rest of the evaluation process expanded to all selected applications and pages would be to high. However, the gathered data of the other applications will be partly investigated during the evaluation. This is necessary to identify the advantages of the new architecture at Xing. As already mentioned, the platform is now composed of several applications instead of one main application. The discussion in the interview leads to the assumption that the process of identifying the source of erroneous behavior might be positively affected by the additional conceptual features of ΘPADx. For example a correlation between two applications could be recognized if the anomaly score for both increases within a short period of time.

Manually Found Anomalies
In the Figures 5.1 – 5.9 the corresponding graphs of all manually found anomalies (MA) are shown. Within the graphs they are marked with red circles. If multiple anomalies are shown in one graph, the anomalies are numbered for a better identification. If possible, a reason for the occurrence of each anomaly is presented as determined in the evaluation interview. The graphs show several non-marked outliers. These were considered as not important enough during the interview. Overall 12 anomalies (M1 - M12) were found. The majority of this set are point anomalies.

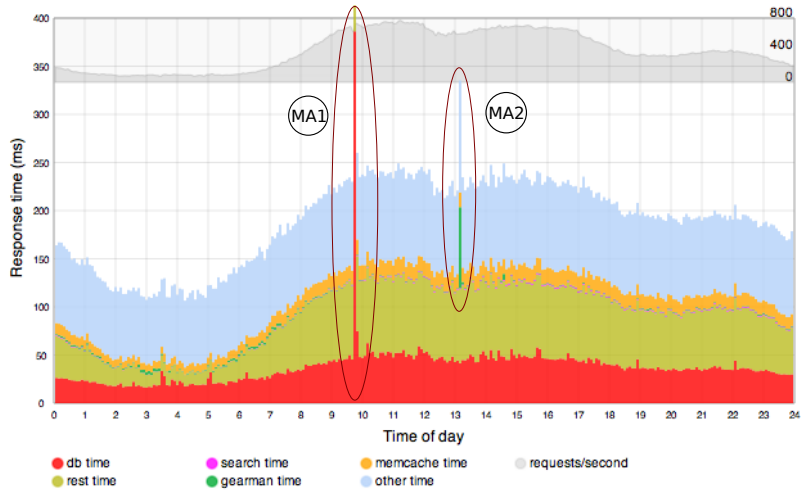

Figure 5.1. (MA1 – MA2) 16.07.2013 09:42 - 09:45 and 13:06 - 13:08
An inefficient database call leads to an increased database time (MA1). A slow Gearman job caused the high gearman time (MA2).

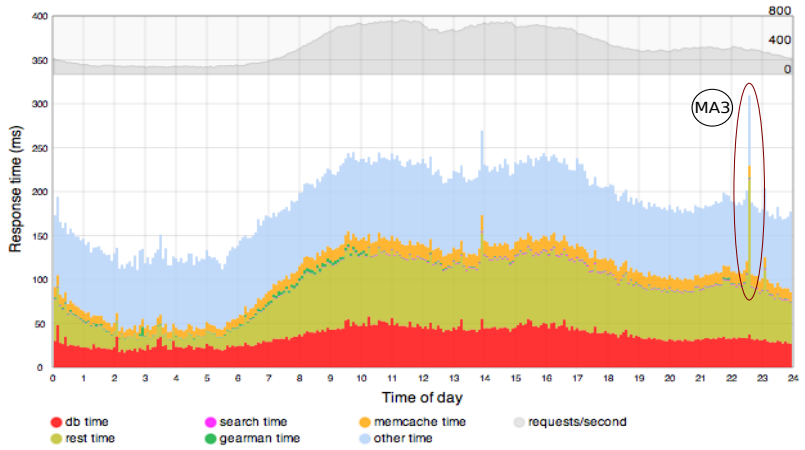

Figure 5.2. (MA3) 17.07.2013 22:30 - 22:32
A slow call to the rest API lead to an increased rest time.

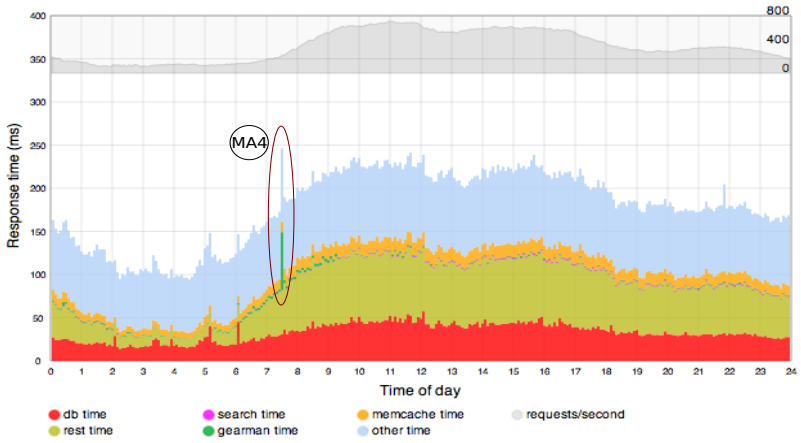

Figure 5.3. (MA4) 18.07.2013 07:27 - 07:28
Again a slow Gearman job is the cause of this anomaly.

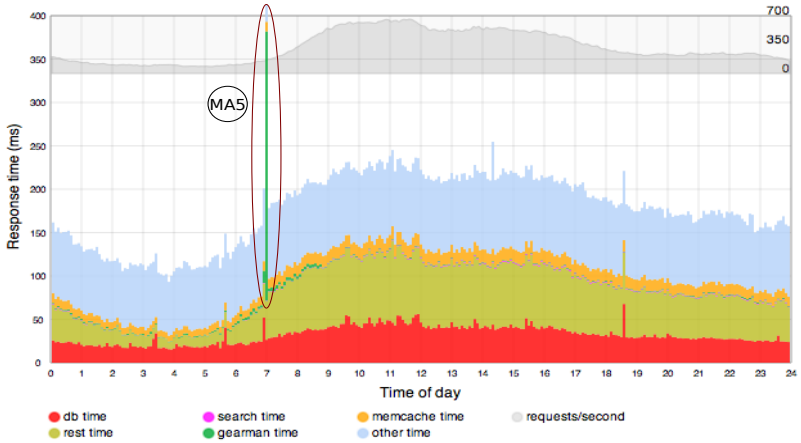

Figure 5.4. (MA5) 19.07.2013 06:56 - 06:58
This anomaly seems to be caused by the same Gearman job as in MA4. This is assumed because of the similar time of appearance.

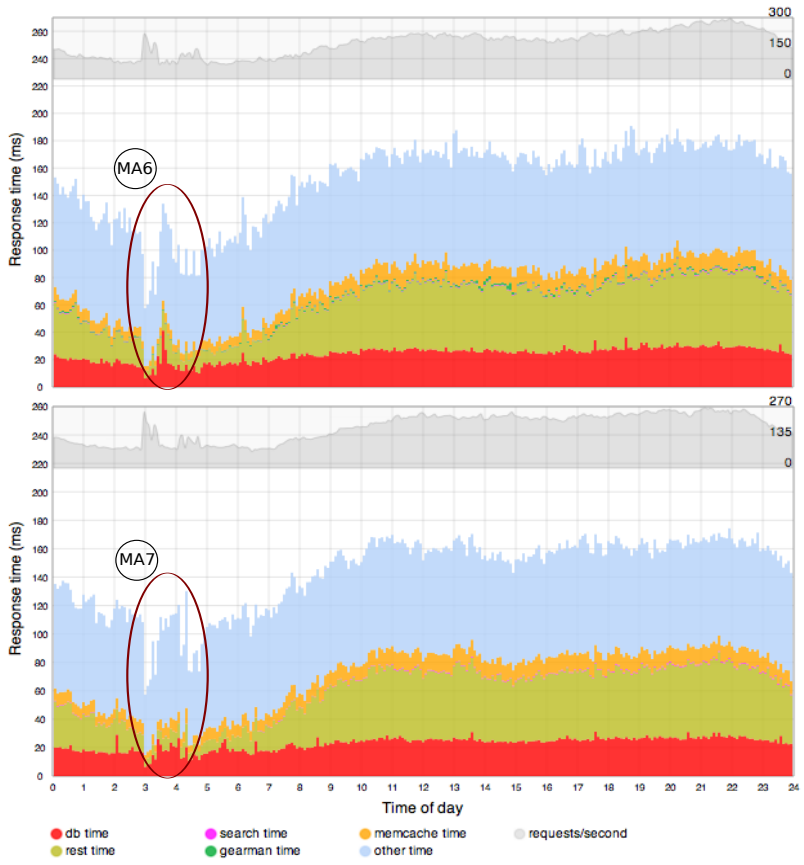

Figure 5.5. (MA6 – MA7) 21.07.2013 and 28.07.2013 02:53 - 04:45
These anomalies are shown in one figure because of their similarity. They are caused by the intrusion detection of the Xing platform that is scheduled for Sundays in the shown time period. This detection sends lot of requests that can be processed very fast. Thus, the total time per request is lower than for usual requests.

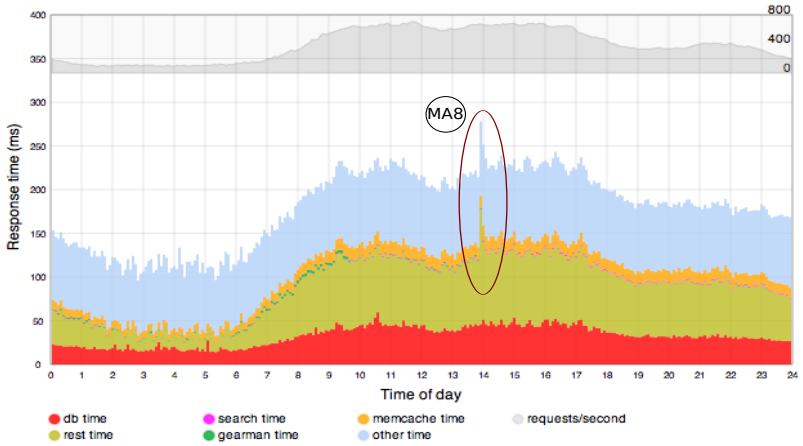

Figure 5.6. (MA8) 24.07.2013 13:54 - 13:55
The reason for this anomaly could not be determined.

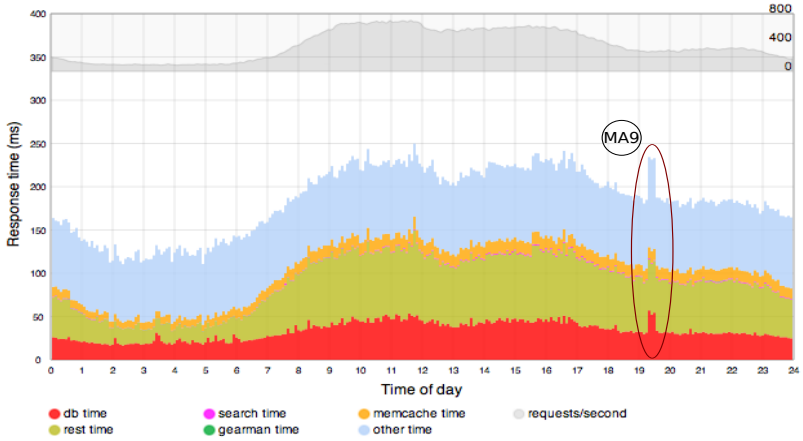

Figure 5.7. (MA9) 25.07.2013 19:15 - 19:29
The reason for this anomaly was a faulty deployment, that caused the performance to decrease. It was reverted after circa 15 minutes.

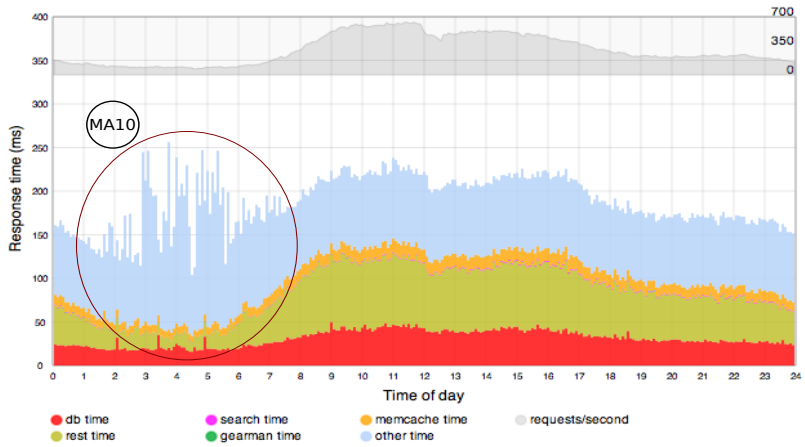

Figure 5.8. (MA10) 26.07.2013 01:30 - 07:20
During the morning hours of this day, the other time keeps to increase from time to time. The reason for this behaviour could not be identified.

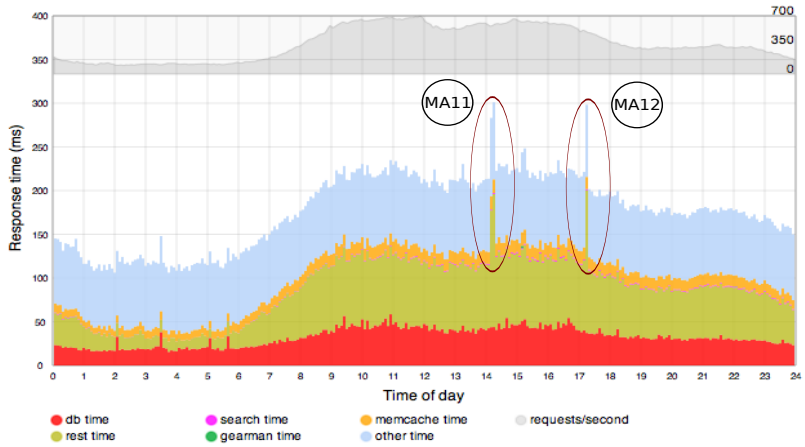

Figure 5.9. (MA11 – MA12) 31.07.2013 14:08 - 14:11 and 17:10 - 17:11
These two anomalies were caused by slow rest calls.

5.4.4 Determine the Configuration

After the data collection is finished a huge sample data set exists. This set offers the opportunity to replay the anomaly detection as often as needed. For the evaluation it is necessary that ΘPADx runs with a configuration that fits the case study system as good as possible, to gain anomaly detection results of high quality. Bielefeld [Bielefeld 2012a]

Table 5.5. This table shows all chosen cases to determine optimized parameters. All cases have to be executed for an increasing value of the threshold. And the complete procedure has to be repeated for ΘPADx with activated pattern checking.

Cases of the ΘPADx Evaluation Configuration			
Case	Forecasting Algorithm	Aggregation Span	TS Window
C1	Mean	30sec	1h
C2	Mean	1min	30min
C3	Mean	1min	1h
C4	Mean	1min	2h
C5	ETS	30sec	1h
C6	ETS	1min	1h
C7	ETS	1min	2h
C8	Arima101	30sec	1h
C9	Arima101	1min	1h
C10	Arima101	1min	2h
C11	SES	1min	1h
C12	SES	1min	2h

already derived the optimal parameters for ΘPAD in the Xing environment. His results will be used as a first indication for the search of a good configuration for ΘPADx with deactivated pattern checking. ΘPADx for example uses different forecasting algorithms and aggregation methods. This results in different combinations of configuration parameters that need to be tested. All cases taken into account are shown in Table 5.5. They are selected because of the following reasons.

First of all, the evaluation should contain all currently available forecasting methods of ΘPADx. Some of the methods have a long runtime. For these, a smaller amount of cases is considered. The used forecasting methods weight past values. Values that are too far in the past only have a small influence on the result. Thus, a too long sliding window is not useful. For the cases, a maximum of 2 hours for the sliding windows is considered. If the aggregation span is set to a value that is too long, the resulting time series is smoothed a lot. This means, that also anomalies may be masked in the resulting time series. That is why the aggregation span of the different cases is settled around one minute.

To detect an anomaly a threshold is used. To find an optimized threshold, the anomaly detection runs several times for all cases. Every time the case is executed, an increased value is used for the threshold. Starting from 0 and increase the threshold up to 1, in 0.01

steps. For every execution all metrics presented in Section 2.2.1 are calculated. At the end of this process an optimized configuration can be found by utilizing the calculated detection comparison metrics.

ΘPADx offers an extension to the anomaly detection approach. Therefore, a comparative evaluation of the results with and without activated extension is targeted. To achieve this, the process of searching the optimal parameters has to be repeated for ΘPADx with activated extension.

5.5 Analysis of the Evaluation Results

In this section the results of the previously presented process of evaluation are analysed. In Section 5.2 the conceptional goals (CG1 - CG4) were defined. We will now step through all goals and see to which extend they are fulfilled.

5.5.1 Determine a Configuration of ΘPADx (CG1)

To determine a good configuration for the case study environment, the opportunity to rerun the anomaly detection on the gathered data is used. A Java application was written that can run the anomaly detection with ΘPADx under a certain configuration on a data set. Whenever the anomaly detection is finished, the results are stored in a database as shown in Section 4.5. The Java application then analysis the anomaly detection result with an increasing value for the anomaly threshold. On this way a good anomaly threshold can be discovered. Table 5.6 shows the calculated detection comparison metrics (see Section 2.2.1) for every previously defined case.

All cases were repeated with activated pattern checking, what led to the same or worse results for all cases. This was expected, because the actual gathered data set contains no collective anomalies that were recognized. The pattern checking is especially designed to improve the anomaly detection results for the appearance of such collective anomalies (see Section 3.2.3) and will here not lead to an improvement. If the pattern checking is able to improve the anomaly detection result in case of collective anomalies, will therefore be clarified in the analysis of CG4.

It can be seen, that the first four cases (C1 – C4) in Table 5.6 all used the *MEAN* forecaster. The table is sorted by the value of the F-measure. Since this value is used to assess the overall quality of the anomaly detection results, this is a first indicator for the *MEAN* forecaster to be the best of the available forecasters for the data of the case study environment. To confirm this indication, ROC curves were created for the best cases of the different forecasters. With their help the trade off between the TPR and FPR can be analyzed.

The best cases per forecaster of Table 5.6 are selected and visualized as ROC curves in Figure 5.10. In the left graph, the blue curve shows the best trade off between TPR and FPR. This curve belongs to the best configuration using the SES forecaster. Since the illustrated curves in the left graph are based on configurations that use different aggregation spans, the trade off between TPR and FPR seems not to be a good indicator to select the best configuration. This is because a lower aggregation span leads to way more anomaly detection results than a higher aggregation span: For example, if the aggregation span is set to 30 seconds, every 30 seconds an anomaly detection is processed. If it is set to 60 seconds, every 60 seconds an anomaly detection is processed and we get only half as much results as we would with a 30 seconds aggregation span. This can lead to different ratios for the TPR and the FPR. However, in the right graph the F-Measures are shown, that were reached by the configurations for every threshold. The best performance is shown by the red curve. Since this curve corresponds to the *MEAN* forecaster this graph confirms the *MEAN* forecaster as the best forecaster for this data set.

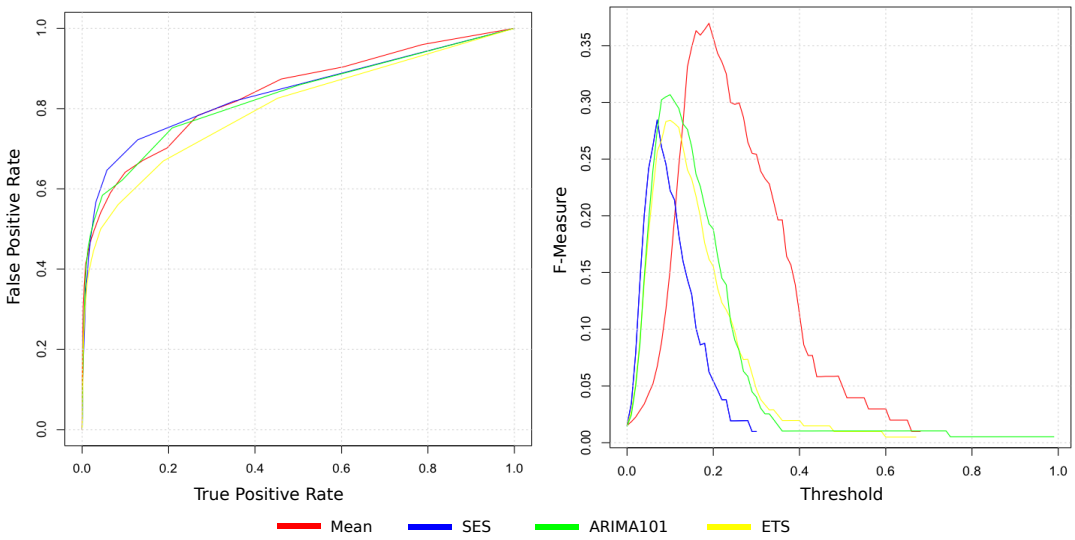

Figure 5.10. ROC curves containing the best found cases per forecast method.

Since the best found configuration is now known, the following questions of CG1 can be answered as follows:

- **Which algorithms to use?**
  The MEAN forecaster shows the best performance.

- **Which threshold detects good?**
  The best found threshold was $\theta = 0.19$.

Table 5.6. The calculated metrics of every case with the best found threshold. The configuration of every case is given in a short form (Forecaster.AggregationSpan.TSWindow). The results are ordered by the F-measure.

Case	Configuration	Best TH	TPR	FPR	Precision	Accuracy	F-measure
C4	MEAN.A1min.W2h	0.19	0.30	0.00	**0.46**	**0.99**	**0.37**
C2	MEAN.A1min.W30min	0.13	0.39	0.01	0.33	0.99	0.36
C1	MEAN.A30sec.W1h	0.15	0.39	0.01	0.31	0.99	0.34
C3	MEAN.1min.W1h	0.17	0.32	0.00	0.37	0.99	0.34
C8	ARIMA101.A30sec.W1h	0.10	0.33	0.00	0.29	0.99	0.31
C5	ETS.A30sec.W1h	0.10	0.28	0.00	0.28	0.99	0.28
C9	ARIMA101.A1min.W1h	0.07	0.31	0.01	0.25	0.99	0.28
C11	SES.A1min.W1h	0.07	0.35	0.01	0.24	0.99	0.28
C7	ETS.A1min.W2h	0.08	0.24	0.00	0.30	0.99	0.27
C12	SES.A1min.W2h	0.08	0.29	0.01	0.25	0.99	0.27
C10	ARIMA101.A1min.W2h	0.07	0.29	0.01	0.24	0.99	0.26
C6	ETS.A1min.W1h	0.07	0.27	0.01	0.21	0.99	0.24

▻ **Which aggregation span is good?**
The aggregations span was set to 1 minute.

▻ **Which sliding window length is good?**
The length of the sliding window was set to 2 hours.

▻ **Which performance attributes matter?**
For an overview of the performance the *total_time* attribute of the application *perlapp* is still the most important performance attribute. An advantage of ΘPADx over ΘPAD is that multiple applications can be processed per instance. Given this opportunity, it is possible to recognize correlations between different applications and therefore between different performance attributes. This involves other performance attributes a lot more than in the single application use case. In Figure 5.11 an example is shown, that was found within the evaluation process. In the *perlapp* applications two anomalies were detected. These anomalies were also identified manually as anomalies, as can be seen in Figure 5.9. Slow rest calls were determined as cause of these anomalies. Therefore, the corresponding application *perlrest* was investigated at the corresponding time points. Since this application shows anomalies that match those of the perlapp application, a clear correlation between the two applications is found. In a common scenario an administrator might see that the anomaly scores of these two applications are high. Since the perlapp application relies on the perlrest application for its rest calls, the cause of the problem can be determined more easily than with data of only a single application.

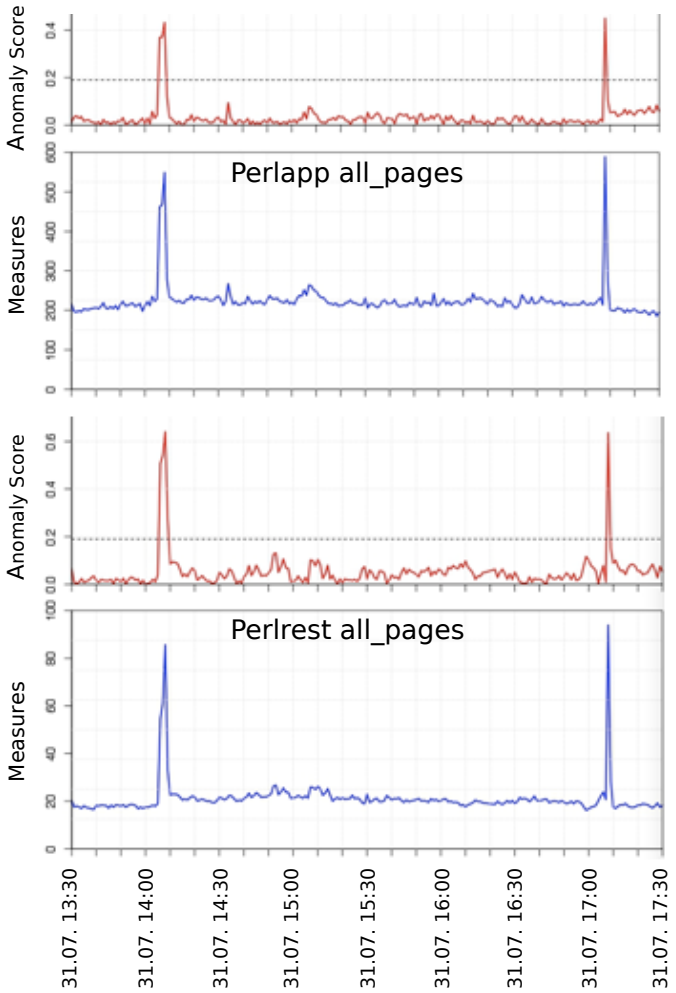

Figure 5.11. Correlation between the *perlapp* application and the *perlrest* application.

5.5.2 The Practicality of ΘPADx (CG2)

Since the approach is designed to find anomalies in live data of software systems, it is necessary that ΘPADx itself is running reliable to constantly processing data. Beside this it is important to clarify, how good the results of ΘPADx are for the use in practice. For this purpose the questions of CG2 are answered with respect to their metrics:

▻ **Is the ΘPADx server stable?**
The ΘPADx implementation ran stable over two weeks of data collection. In this phase ΘPADx was configured to process the anomaly detection on live data and additionally

to store the raw data. Based on this, the ΘPADx implementation can be stated as stable for the present. However, further observation of the stability of ΘPADx during the use in practice over a longer duration should be planned. During the data collection phase the three most important processes java, mongod and Rserv-bin.so reached constantly low values for the CPU utilization. Only the java process got peaks every time the aggregation span is over. At this time the aggregation and forecasting methods are processed, which leads to more calculation effort.

- ▻ java: idle $0.5\% - 3.5\%$, peaks up to 20%
- ▻ mongod: $0.5\% - 5\%$
- ▻ Rserv-bin.so: $0.0\% - 0.5\%$

ΘPADx starts with a low memory usage around $26mb$ and then consumes more memory. This is caused by the empty sliding windows in the beginning, that get filled in the first period of the run time. As soon as the sliding windows are filled, the memory usage is about to be constant. For the data collection phase, ΘPADx consumes around $73mb$ of memory. In this case ΘPADx was configured to work on the data of 14 different applications. The memory consumption should be smaller for a lesser amount of applications. Since the memory consumption is about to be constant, no memory leaks were found that may influence the stability of the implementation during a long term execution.

- ▻ **How precise is the detection?**
  Under the best selected configuration the anomaly detection of ΘPADx reached a value of $PREC = 0.46$. This means that there are slightly more false warnings given than correct detected actual anomalies.

- ▻ **How accurate is the detection?**
  For the same configuration a accuracy of $ACC = 0.99$ was reached. This value indicates that the ratio between the correct detections to all made detections is high. Based on this fact, an anomaly detected by ΘPADx is likely to be an actual anomaly.

- ▻ **How good is the overall quality of the detection results?**
  The approach reached a value of $F - measure = 0.37$. Since the F-measure is a value that describes the over all quality of a detection approach, this seems like a fairly bad result for the ΘPADx approach.

  However, these metrics do not cover the following points. Some anomalies just occur for a few seconds, while others appear over more than an hour. There can also be variations in the interval of an anomaly, where the measures toggle from abnormal values to normal values from time to time. But these intervals will overall still be marked as anomaly. Additionally, the manually found anomalies were defined in intervals of a one minute granularity. This means that anomalies may extend over the edges of these

intervals, which is hard to interpret for the evaluation script. For the approach, to reach high precision, accuracy or F-measure values, every time point in an interval needs to exactly fit to the manually determined anomaly intervals.

For the usage of ΘPADx in practice, the following fact is more important. The approach should at least recognize an anomaly for one time and as soon as possible. This will for example offer the opportunity to alert an admin, who can pull out a quick reaction on the anomaly. For this practical scenario it is not necessary to detect all values of the anomaly interval correctly. From the analysis of different anomaly detection graphs, that were sighted during the evaluation phase, this scenario can be found to be fulfilled by ΘPADx.

5.5.3 Anomaly Causality Research (CG3)

Anomalies can be caused by many different factors. Within the evaluation several reasons for appearing anomalies in the actual data collection were identified within the evaluation interview. In the following the found causes are presented and a brief comparison to the causes found by Bielefeld [Bielefeld 2012a] is made.

- **Which types of real anomalies occurred?**
  For the actual data collection these kinds of anomalies occurred:
  - Slow database call (MA1)
  - Slow Gearman job (MA2, MA4, MA5)
  - Slow call to the REST API (MA3, MA11, MA12)
  - Intrusion Detection (MA6, MA7)
  - Faulty deployment (MA9)
  - Could not be determined (MA8, MA10)

Over all, five different causes of anomalies were found. For two anomalies no exact reason could be determined. The actually found causes lead to point anomalies in the most cases. Slow calls to the REST API or the database for example only affect a few users and also only under certain circumstances. In the data set collected by Bielefeld some factors were found that have a stronger influence on the system. For example the complete disconnections of the logjam servers or server faults. Since it takes more time to fix such issues this leads to more collective anomalies.

5.5.4 Assess the Extensions (CG4)

- **Are the additional conceptual features useful?**
  The goals G3.2 and G3.3 describe two conceptional extensions of the ΘPAD approach.

Table 5.7. All calculated metrics for the runs of the back-to-back test. (B1 = original configuration of Bielefeld, B2 = corrected anomaly intervals, B3 = corrected anomaly intervals and activated pattern checking)

Run	TP	FP	FN	TN	TPR	FPR	PREC	ACC	F-M
B1	38	239	183	16815	0.17	0.01	0.14	0.97	0.15
B2	81	196	140	16858	0.37	0.01	0.29	0.98	0.32
B3	81	142	140	16912	0.36	0.01	0.31	0.98	0.36

As discussed in the Chapters 3 and 4, the corresponding features are realized within ΘPADx. Therefore, ΘPADx is now able to process the data of different applications and for different metrics as well.

In the analysis of CG1 (see Section 5.5.1) a useful use case for these features was already found. A correlation between the data of two interrelated applications was identified. This led to a faster identification of the cause of an appeared anomaly. Since it is possible to use such correlations in practice to make it more easy to find the cause of an anomaly, these features are classified to be useful.

Beside this, the conceptual extensions offer an additional advantage. Usual software systems consist of several applications. This is also the case for the case study environment at Xing. With ΘPAD it is only possible to collect the data of a single application out of these. With ΘPADx now the anomaly detection can be processed for all applications or, at least, for the most important applications. This leads to a better system coverage and more useful data for visualizations or alerts, for example in the case of a system dashboard.

▷ **How differs the quality of the detection results?**
The main extension to the ΘPAD approach is the pattern checking (see Section 3.2.3). This extension aims to reduce the number of false positives that were produced by ΘPAD in connection with collective anomalies. Since the actual data collection did not contain collective anomalies, the data collection of Bielefeld [Bielefeld 2012a] is used to assess the pattern checking extension.

In Section 5.4.1, that describes the back-to-back test, three different runs of the test (B1, B2, B3) were already mentioned. B1 equals the configuration of Bielefeld, B2 takes corrections of the time stamps in consideration and B3 additionally used the pattern checking extension. The calculated comparison metrics of these runs are shown in Table 5.7. As expected, the run of B1 offered the same results which were found within the evaluation phase of Bielefeld [Bielefeld 2012a]. It can be seen, that with the correction of the time stamps (B2), the TPR, the PREC and the F-Measure reached results that were more than twice as good as the results of B1. This means, that the ΘPAD approach is

not as bad as described by Bielefeld. However, the activation of the pattern checking should still show better results, because the time stamp correction is not correlated with the mentioned problem of additional false positives.

The results of B3 seem to have just small deviations from the results of B2. The F-measure of B3 for example is about 0.04 higher than the F-measure of B2. However, the most important value for this extension is the amount of false positives (FP). As can be seen, a reduction of 54 FP from 196 FP for B2 down to 142 FP for B3 was possible without negative influence on the values of TP and FN. This result is supported by Figure 5.12. At the top, the results for B2 can be seen, while the graphs below show the results for B3. The blue line marks the measures, the green line marks the forecasts and the red line marks the anomaly scores. In the results of B2 it can be seen, that the forecasts stay high, even if the anomaly is already over (1). This is caused by a problem with the sliding window, that is filled with abnormal values during the time period of the anomaly, as described in Section 3.2.3. These high forecast values deviate strongly from the measurements, that return to the normal level after the anomaly is over. This results in high anomaly scores (2). In the graphs of B3 it can be seen, that the forecasts are falling way faster after the end of the anomaly (3). As conclusion, the amount of detected anomalies after the end of the actual anomaly is lower. This is were the pattern checking is reducing the amount of FP in difference to the ΘPAD approach. For the use in practice, this results in lesser fail alerts, that would else waste the time of an administrator.

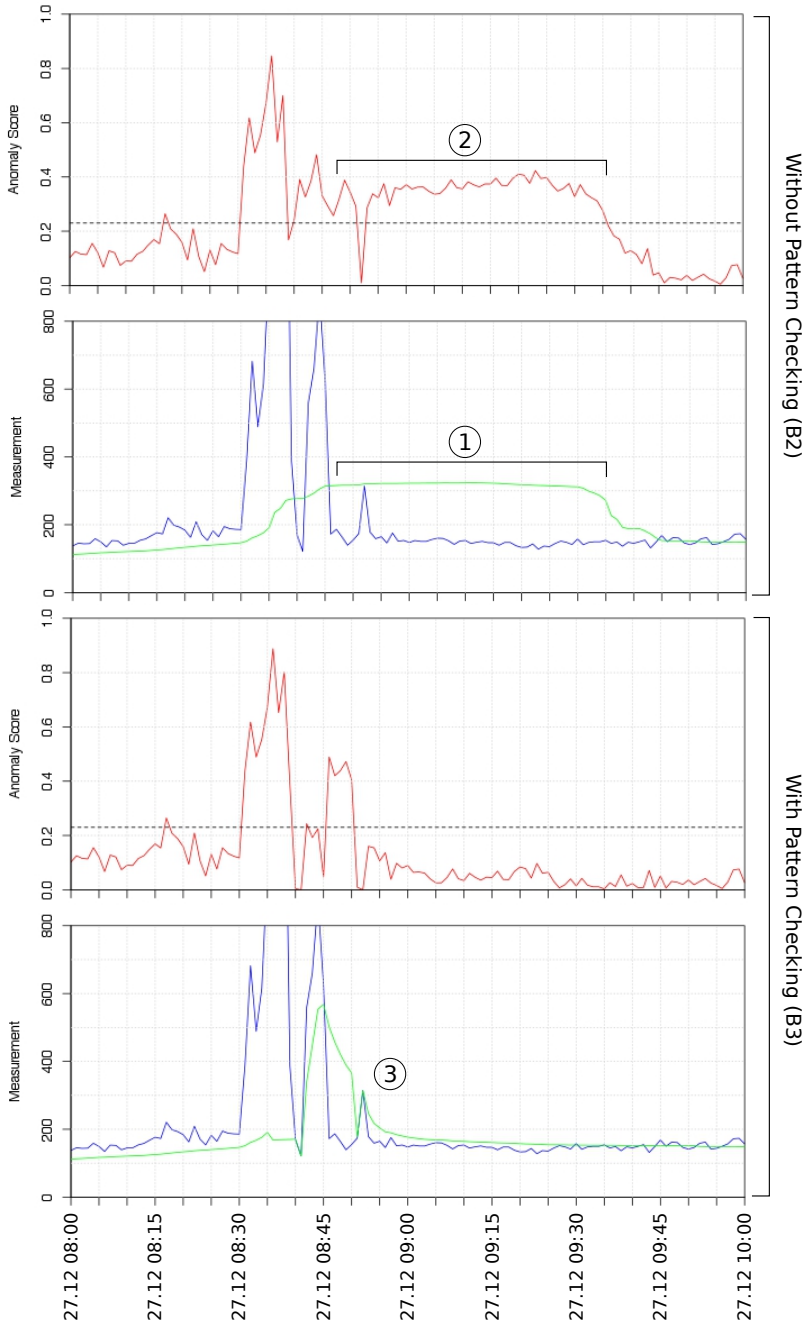

Figure 5.12. These graphs show the difference between the anomaly detection approach with and without activated pattern checking in case of the occurrence of an collective anomaly.

Chapter 6

Related Work

Anomaly detection is a problem that is applicable to many domains. Due to this fact, a multitude of works can be found that cover anomaly detection for a specific research field. In this chapter a list of tools and research is presented that is related to the problem field of ΘPADx, the forecasting and anomaly detection on performance data. Every mentioned product or approach is briefly introduced and compared to ΘPADx.

6.1 Combining Time Series Models for Forecasting

Often statistical models, as for example ARIMA models, are used to predict future values on time series data. A certain model must be selected that fits to the time series, that is going to be processed. However, over the duration of a time series, the selected model may not always fit the actual behavior. Therefore, Zoua and Yangb [Zou and Yang 2004] developed an algorithm, called *AFTER*, that is able to combine several models.

In situations where it is hard to identify one best-fitting statistical model, the AFTER approach reached better results than a static model selection approach.

For the ΘPADx approach this can be used to improve the prediction performance for environments, where the ARIMA forecaster is part of the best configuration. Overall, this approach shows that the combination of several models or methods may increase the quality of the corresponding forecasting results, just as similarly executed by ΘPADx.

6.2 Modeling Multiple Time Series for Anomaly Detection

The approach, introduced by Chan and Mahoney [Chan and Mahoney 2005] is able to generate a reference model of a system with the help of multiple corresponding training time series. Three algorithms were developed, that combine the information of the training time series in a so-called box model. This box model can then be used as reference model for predictions of the system's data.

The approach was evaluated on a data set of the NASA that contains sensor data of several valves. It was found that the approach performed better than two common box

modeling approaches.

The working method of this approach is similar to ΘPADx: a reference model is generated and deviations between the model and the actual behavior are calculated. In difference to ΘPADx, the approach of Chan and Mahoney needs training sets to be able to work. A certain amount of time must therefore be scheduled, before the anomaly detection could start.

6.3 Workload Aware System Monitoring Using Performance Predictions Applied to a Large-Scale E-mail System

For the most anomaly detection approaches a fixed threshold is used to determine whether a system is in a normal or abnormal state. It is known, that the performance data of a system is highly related to the actual workload. Fixed thresholds are often configured to work in the case of workload peaks. This may lead to slow recognition times of an abnormal system behavior, especially in low workload situations. Therefore, Rathfelder et al. [Rathfelder et al. 2012] developed an approach, that dynamically adjust the threshold for the anomaly detection in relation to the actual workload. For the reference model generation the software system is modeled with the help of the *Palladio Component Model* (PCM), and performance simulations on this model are used as predictions.

The approach was tested on the e-mail system of a large e-mail provider from Germany and reached an error rate of mostly less than 10% for the prediction of the systems resource utilization.

ΘPADx is working with fixed anomaly thresholds as well. For the anomaly detection with ΘPADx, especially on systems in which the performance data underlies huge workload variations, this methods seems to be an appropriate way to improve the quality of the anomaly detection.

6.4 Workload Classification and Forecasting

Herbst [Herbst 2012] presented an approach, called *WorkloadClassificationAndForecasting* (WCF), that is able to use the actual workload intensity behavior to improve its forecasting results. A forecasting method is selected, that is known to offer results of high quality under the actual workload intensity pattern. To select an appropriate forecasting method, the accuracy of several forecasting strategies is evaluated and incorporated into a decision tree. The decision tree considers forecasting objectives to finally select the best fitting forecasting method.

The WCF approach was applied to real world data and the relative error metric was used to compare the approach with the static application of the extended exponential smoothing method. The approach of Herbst reached a 63% reduced relative error. In further case studies it was found that the WCF approach is able to prevent between 52% and 70% of the violations of a given service level agreement.

The WCF approach selects different forecasting methods in relation to the actual processed data, a similarity to ΘPADx. While the WCF approach uses the workload intensity behavior to choose a forecaster, the ΘPADx approach considers the type of an anomaly for the decision to use or not to use the pattern checking forecaster. However, the WCF approach can be used to improve the forecasting results in general and is therefore considered to be integrated into ΘPADx.

6.5 Online System Problem Detection by Mining Patterns of Console Logs

Xu et al. [Xu et al. 2009] developed a system for the online problem detection of software systems with console logs as input. In a first stage the approach is frequently mining pattern of the console logs. The pattern are valued related to the pattern distribution. The most dominant pattern, which are pattern that appear often, are valued as normal. In a second stage, less dominant pattern are analyzed based on principal component analysis (PCA) methods.

The approach was evaluated with the help 24 million lines of log messages from a large-scale real world application. It was shown, that the approach matches or outperforms even the most common offline analysis methods.

The approach of Xu et al. shows the strongest similarities to ΘPADx. Although their approach operates on console logs, they are still offering online analysis and are using two different phases to increase the accuracy of the approach.

6.6 An Automated Approach to Forecasting QoS Attributes Based on Linear and Non-linear Time Series Modeling

A common method to predict future values, in the case of Amin et al. [Amin et al. 2012b] Quality of Service attributes, is provided by ARIMA models (see Section 2.1.3). Such data is often assumed to be linear and therefore a linear ARIMA model is selected for the predictions. Amin et al. analyzed real world QoS data sets and found that the data often shows non-linear behavior and is therefore hard to predict with existing ARIMA models.

Because of this, they developed an approach that automatically adjusts the underlying ARIMA model through integration to improve the predictions.

The approach was evaluated on several real world data sets and reached a forecasting accuracy increased by an average of 35.4% in comparison to common ARIMA models.

ΘPADx can be configured to use the ARIMA101 forecaster, which is one of the common ARIMA models. For non-linear data sets this approach could be considered as a further improvement for the prediction accuracy of ΘPADx.

6.7 Statistical Detection of QoS Violations Based on CUSUM Control Charts

This approach, called *Automated Detection of QoS Attributes Violations* (AuDeQAV) and also developed by Amin et al. [Amin et al. 2012a], detects QoS violations based on descriptive statistical methods. First, QoS-related statistics are collected at system runtime to describe the systems normal behavior. The statistics are used to build a CUSUM control chart. For each new QoS observation the CUSUM control chart is updated and checked for QoS violations.

In an evaluation phase it was found that AuDeQAV is able to outperform simple threshold based approaches.

In difference to ΘPADx, AuDeQAV uses descriptive statistical methods instead of forecasting methods to generate a reference model.

6.8 Anomaly Detective

The *Anomaly Detective*[1] tool, developed by prelert, provides anomaly detection based on machine learning algorithms. It is used as plugin for the monitoring and analysis software Splunk[2]. Due to the usage of machine learning, prelert promises that the tool is 100% self-learning and therefore releases the user from configuration work. The only prerequisite is the connection to the data sources. Similar to ΘPADx, the tool works online and is able to work on different data sources and metrics. Additionally, it provides an extensive amount of visualizations for the analysis results by using the visualization interface of Splunk. A free version is available that supports 0.5GB indexed data per day and does not contain the real-time feature. The full feature premium version is priced according to the daily indexed data volume.

[1] http://www.prelert.com/products/anomaly-detective.html
[2] http://www.splunk.com

6.9 STATISTICA Advanced

STATISTICA Advanced[3] from the manufacturer StatSoft is a tool that contains a set of data analytic features. In the advanced version the tool contains a module for time series forecastings based on the common forecasting methods that are also used by ΘPADx. The tool offers the opportunity for several analyses and transformations of the time series data, for example component model decomposition, smoothing methods or cross correlation analysis. Additionally, the time series can be stored for later usage and a log of all made transformations on the time series is maintained by the tool. For visualization purposes the tool offers a wide range of different plots. The pricing for the tool is handled individually per request.

6.10 New Relic

New Relic[4] offers a monitoring service for web and mobile applications. In contrast to the already mentioned anomaly detection tools, the services of New Relic are offered as SaaS. The tool is able to analyze performance data for example real-time browser performance data, transaction performance data, database related performance data and server performance data like CPU or RAM utilization. Besides the monitoring, alerting is supported as well. The alerting is realized via configurable thresholds, as known from ΘPADx. New Relic takes 24$ per month and server for the standard version and 149$ per month and server for the pro version.

[3] http://www.statsoft.com/products/statistica/advanced/
[4] http://newrelic.com

Chapter 7

Conclusion

This chapter will cover a brief summary of this thesis in Section 7.1. In the beginning of this document the organizational goals were formulated. In Section 7.2 these goals are revisited and a closing assessment, whether the goals are reached or not, takes place. Finally, ideas for further improvements of the approach and additional planned features for the ΘPADx implementation are presented.

7.1 Summary

The introduction pointed out that monitoring, especially performance monitoring, is becoming more and more important. This is caused by the complexity large-scale software systems can reach, what makes it hard to keep track of the systems health. Whenever issues are affecting the systems behavior, these issues may manifest themselves as anomalies in the systems performance data. Despite the size and the complexity the underlying software system may have, a fast reaction to these anomalies is necessary to prevent slow response times or even the unavailability of the system. This is why an online performance anomaly detection (ΘPAD) approach was developed by Bielefeld [Bielefeld 2012a]. However, the results of ΘPAD were negatively influenced whenever anomalies occur over a long time period and usability improvements were necessary for the corresponding implementation. Due to this facts, an approach called ΘPADx, that is based on ΘPAD but extends the existing approach, was motivated.

The ΘPADx approach is based on several mathematical definitions, concepts and technologies. Therefore, Chapter 2 introduced the mathematical foundations of time series analysis and defined the most important terms in the domain of anomaly detection. Also the key technologies used for the implementation, the case study environment and the working method of ΘPADx were presented.

With the knowledge about the working method of ΘPAD an analysis of the approaches weaknesses was made. It turned out, that the used forecasting methods, which are part of the anomaly detection process of ΘPAD, offer results of poor quality, whenever a long term anomaly, also called collective anomaly, occurs. Based on this analysis the ΘPAD approach was extended and now called ΘPADx. ΘPADx uses an additional forecasting method called

pattern checking, whenever an collective anomaly is detected. This forecasting method makes use of data from a long term storage of performance data and takes advantage of existing seasonal pattern.

In the following, a corresponding implementation of ΘPADx was developed. This implementation includes the extension and several conceptual changes of the approach. ΘPADx is now capable to process anomaly detection for several applications and metrics per instance, what was not the case for ΘPAD.

To assess the extension of the approach, the implementation was evaluated. Xing, a social network platform focused on business contacts, served as case study environment. The implementation was tested over a period of two weeks and the results were analyzed and compared to the performance of ΘPAD. The results showed an improved quality of the anomaly detection results and the conceptual changes were found to increase the usability of the approach and to be helpful for a faster determination of anomaly sources.

7.2 Goals Revisited

In Chapter 1 four organizational goals were defined. In the following, these goals are revisited and a final assessment is given, whether the goals are fulfilled or not.

7.2.1 G1: Improvement of the ΘPAD anomaly detection approach

The most important goal was to improve the anomaly detection approach. Improvements can be made in certain ways. G3 for example aims on several conceptional changes of the approach and the corresponding implementation. Since ΘPAD and ΘPADx are delivering anomaly detection results, these results can be of a varying quality. This goal (G1), was primarily defined to address a quality increase of the anomaly detection results.

Due to the extensive evaluation phase Bielefeld [Bielefeld 2012a] executed in his thesis, it was possible to compare ΘPAD with ΘPADx. The comparison metrics offered an improved anomaly detection quality. While the F-Measure only increased from 0.32 to 0.36, the improvement reduced the amount of false positives from 196 to 142. These are more than 50 false alarms that were eliminated on the test data Bielefeld collected during his evaluation. This helps to make the approach more useful in practice, since false alerts may cause an unnecessary action by an administrator. A F-Measure of 0.36 seems to point out a weak anomaly detection approach. But during the evaluation it was found that several factors have negative influence on the comparison values. A detailed view on these issues was given in Section 5.5.

For the actual data collection presented in Section 5.4.2, the ΘPADx approach reached an

F-Measure of 0.37, which is similar to the results on the data of Bielefeld [Bielefeld 2012a]. The evaluation revealed that the improvements are not applicable to the data of the actual data collection, because it did not include collective anomalies. Overall, ΘPADx gives at least one alert for 11 out of 12 anomaly intervals determined for the actual data collection. Since in practice it is not needed to recognize an anomaly over the complete duration, this qualifies the approach for the usage in production.

7.2.2 G2: Implementation of the ΘPADx approach

For the use of the approach in practice an implementation, containing all developed extensions, was necessary. The implementation is based on a reworked version of ΘPAD, following the pipes-and-filters plugin structure of the Kieker framework. The developed extensions were integrated in the implementation in the form of additional and reworked filters, an additional forecasting method, a new adapter and new record types (see Chapter 4).

During the evaluation phase performance data of Xing was collected over a duration of more than two weeks. For the ΘPADx implementation, as well as for all related processes and libraries, neither a downtime nor errors were recognized. The measures for CPU utilization and memory consumption showed constant values as well, what indicates no memory leaks or similar common problems. Due to these facts the implementation is considered to be stable and ready for use in practice.

7.2.3 G3: Integration of the extended approach in the redefined Xing environment

Several conceptional changes were related with this goal. First of all, Xing used a different messaging technology to deliver the performance data, called ZeroMQ (see Section 2.4.3). An adapter was developed, that enables ΘPADx to gather the performance data via ZeroMQ and to deliver the data to the ΘPADx Kieker plugins. This proved the adapter concept, already chosen by Bielefeld [Bielefeld 2012a], to be flexible and to enable the usage of ΘPADx in a variety of environments.

Beside this, Xing decides to split the main application, that was processed by ΘPAD, into several interrelated applications. Therefore, ΘPADx now must be able to process the data of several applications. Additionally, it should be possible to optionally process multiple metrics per application (see Section 2.3), while ΘPAD was developed to process only the total time. To achieve this, the filters of ΘPAD are extended to be able to load and store data according to an unique identifier, as described in Section 4.4.2. With the help of this feature, ΘPADx was able to process the anomaly detection for 14 different Xing applications during the data collection phase. Since software systems often consist of several applications and several performance data is monitored, these features of ΘPADx can be used to reach a better system coverage and to find correlations in the performance

data of the applications.

Summing up the mentioned advantages, the conceptual changes are found to increase the usability of ΘPADx.

7.2.4 G4: Evaluation

The last defined goal was to execute an evaluation, to be able to assess the quality of the anomaly detection results of ΘPADx and to compare them with the results ΘPAD reached. The already mentioned implementation of the ΘPADx approach was the most important prerequisite for the evaluation. With the help of an actual data collection and the data collected by Bielefed [Bielefeld 2012a], an assessment of the ΘPADx approach and a comparison with ΘPAD was successfully executed. The assessment was supported by an evaluation script. It was used to determine configurations that fit to the gathered performance data and to calculate the detection comparison metrics (see 2.2.1). Additionally, the visualization with *R* helped to find crucial parts in the performance data and in the anomaly detection results.

7.3 Future Work

During the development of ΘPADx several ideas and additional features were considered to be useful. Because of time limitations, some of these are not included in the current state of the implementation. In this section these ideas and features are briefly presented. They are considered for future implementation upgrades or following theses.

7.3.1 Improved Configuration

The possibility to process anomaly detection for multiple applications is a goal that is realized with the implementation of ΘPADx. ΘPADx now allows to define a range of identifiers, for example the names of the applications. For every identifier, the data is processed without messing it up with the data assigned to an other identifier. While this is a clear separation of the applications performance data, they currently still share the same configuration. To be able, to generate more accurate results for all observed applications, a separation of the configuration is necessary as well. This would improve the anomaly detection results per application and would also make correlations between different applications more accurate.

The current application filters already load data in dependence on a particular identifier, as described in Section 4.4.2. For a future implementation it might be thinkable to additionally load a configuration file, that stores the configuration for the actual identifier.

7.3.2 Automatic Correlation Detection

As already mentioned in the last section, it is possible to process performance data for multiple applications. This was used during the evaluation of the ΘPADx approach, where the data of 14 different applications was processed. With the help of visualizations of the evaluation results, correlations were found between different applications. An example for this was presented in Figure 5.11, where the perlapp application showed abnormal behavior in the same time intervals as the perlrest application. With knowledge of the underlying system, the perlrest application was identified as the source of the abnormal behavior, since the perlapp application was calling the perlrest application right in these intervals.

However, to find such correlations in the current state of the implementation, the continuous observation of the visualized anomaly detection results, for example by an admin, would be necessary. For further improvements of the implementation, therefore an automatic correlation detection approach could be chosen. A possible example for the calculation of an correlation score is given by Ide et al. [Ide et al. 2007], where the correlation between the data of several car sensors is calculated with the help of stochastic nearest neighbors.

7.3.3 Automatic Pattern Recognition

As an additional forecasting algorithm, the pattern checking forecaster was introduced. To achieve high quality results with this forecaster, a proper configuration is necessary. For the pattern checking forecaster, this includes knowledge about the seasonal component, that influences the performance data of the selected software systems. For example at Xing, the performance data had a similar behavior in intervals of one week. This seasonal pattern was easily determined by observing the performance data over several weeks.

An automatic approach, that is able to recognize such seasonal pattern, offers several advantages. At first, it lowers the configuration needs for the forecasting method and therefore also the amount of knowledge the user needs of the selected software systems. Beside this, an automatic approach enables the opportunity to readjust the seasonal pattern automatically. This can be interesting for time spans, in which the usual workload of the system differs. As an example, we could take the weeks right before Christmas, when the most online shops have an increasing amount of orders. An automatic readjustment could help in such situations to prevent the quality of the pattern checking results from decreasing. Since a connection to *R* is used already, a starting point for the implementation of an automatic pattern recognition could be the `decompose` function of *R*.

Chapter 8

Acknowledgments

I wish to thank everyone who helped me complete this thesis:

- Prof. Dr. Wilhelm Hasselbring
  For creating the interesting topic and bringing me in contact with the XING AG.
- Dipl.-Inf. Jan Waller and Dipl.-Inform. André van Hoorn
  For being always available and helpful supervisors and for giving all the advises and ideas that helped me to improve the thesis.
- Dr. Stefan Kaes
  For all the help he offered regarding to the thesis topic and the Xing environment. Additionally, for all the insights he gave about performance related data, monitoring aspects and the Xing platform.
- XING AG
  For being an awesome company that gave me permission to the best thinkable resources to complete my thesis and for letting me work in their environment. A special thanks to the Architecture Team, that welcomed me friendly and granted some insights of a software architects everyday work.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel,

Bibliography

[Amin et al. 2012a] Ayman Amin, Alan Colman, and Lars Grunske. "Statistical detection of qos violations based on cusum control charts". In: *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. 2012, pages 97–108.

[Amin et al. 2012b] Ayman Amin, Lars Grunske, and Alan Colman. "An automated approach to forecasting qos attributes based on linear and non-linear time series modeling". In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. 2012, pages 130–139.

[Basili et al. 1994] Victor R. Basili, Gianluigi Caldiera, and H Dieter Rombach. "Goal question metric paradigm". In: *Encyclopedia of Software Engeneering Volume 1*. 1994, pages 528–532.

[Becker et al. 2006] Steffen Becker et al. Trustworthy Software Systems: A Discussion of Basic Concepts and Terminology. Carl-von-Ossietzky University of Oldenburg, 2006.

[Bell 1994] Richard A. Becker. A Brief History of S. AT & T Bell Laboratories, 1994.

[Bielefeld 2012a] Tillmann Carlos Bielefeld. Online Performance Anomaly Detection. 2012.

[Bielefeld 2012b] Tillmann Carlos Bielefeld. OPAD: Online Performance Anomaly Detection with Kieker. KoSSE-Symposium Application Performance Management (Kieker Days 2012), 2012.

[Box and Jenkins 1990] G. E. P. Box and G. Jenkins. Time Series Analysis, Forecasting and Control. Holden-Day, Incorporated, 1990.

[Chan and Mahoney 2005] Philip K. Chan and Matthew V. Mahoney. "Modeling multiple time series for anomaly detection". In: *Fifth IEEE International Conference on Data Mining*. IEEE Computer Society, 2005, pages 90–97.

[Chandula et al. 2009] Varun Chandula, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: a survey". In: *ACM Comput. Surv. Volume 41*. 2009, 15:1–15:58.

[Chodorow 2011] Kristina Chodorow. Scaling MongoDB: Sharding, Cluster Setup, and Administration. O'Reilly Media, 2011.

[Facebook 2013] Facebook, 2013. URL: http://newsroom.fb.com/Timeline.

[Fawcett 2006] Tom Fawcett. "An introduction to roc analysis". In: *Pattern Recognition Letters - Special Issue: ROC Analysis in Pattern, Volume 27 Issue 8*. June 2006, pages 861–874.

[Frank et al. 2001] R.J. Frank, N. Davey, and S.P. Hunt. "Time series prediction and neural networks". In: *Journal of Intelligent and Robotic Systems Volume 31*. Kluwer Academic Publishers, 2001, pages 91–103.

[Gamma et al. 1995] Erich Gamma et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., 1995.

[GrayLog2] Official Graylogl Website. URL: http://graylog2.org/.

[Gustavsson and Andler 2002] Sanny Gustavsson and Sten F. Andler. "Self-stabilization and eventual consistency in replicated real-time databases". In: *Proceedings of the first workshop on Self-healing systems*. 2002, pages 105–107.

[Herbst 2012] Nikolas Roman Herbst. Workload Classification and Forecasting. Diploma Thesis. Karlsruhe Institute of Technology (KIT), 2012.

[Herbst et al. 2013] Nikolas Roman Herbst et al. "Self-adaptive workload classification and forecasting for proactive resource provisioning". In: *4th ACM SPEC International Conference on Performance Engineering (ICPE 2013)*. Apr. 2013.

[Hyndman 2013] Rob J. Hyndman. Forecasting Functions for Time Series. 2013. URL: http://cran.r-project.org/web/packages/forecast/forecast.pdf.

[Ide et al. 2007] T. Ide, S. Papadimitriou, and M. Vlachos. "Computing correlation anomaly scores using stochastic nearest neighbors". In: *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. 2007, pages 523–528.

[Ju 2010] Jiehui Ju. "Research on key technology in saas". In: *Intelligent Computing and Cognitive Informatics (ICICCI)*. Zhejiang University of Science and Technology, June 2010, pages 384–387.

[Kaes 2013] Stefan Kaes. Github Repository and Wiki for Logjam. 2013. URL: https://github.com/skaes/logjam.

[Kedia et al. 2005] Vipul Kedia, Vamsidhar Thummala, and Kamalakar Karlapalem. "Time series forecasting through clustering - a case study". In: *International Conference on Management of Data (COMAD) Volume 11*. Jan. 2005.

[Kieker Project 2013] Kieker 1.6 User Guide. Kieker Project, Germany, 2013. URL: http://kieker-monitoring.net/documentation/.

[Koziolek 2008] Heiko Koziolek. "'Dependability metrics'". In: Springer-Verlag, 2008. Chapter Introduction to performance metrics, pages 199–203.

[Lazarevic et al. 2003] Aleksandar Lazarevic et al. "A comperative study of anomaly detection schemes in network intrusion detection". In: *3rd SIAM International Conference on Data Mining*. 2003, pages 25–36.

[Limam 2010] Noura Limam. "Assessing software service quality and trustworthiness at selection time". In: *IEEE Transactions on Software Engineering*. IEEE Computer Society, Jan. 2010, pages 559–574.

[Maxion and Roberts 2004] Roy A. Maxion and Rachel R. Roberts. "Proper use of roc curves in intrusion/anomaly detection". In: *Technical Report Series no. CS-TR-871*. University of Newcastle upon Tyne. Computing Science. 2004.

[Miniwatts 2013] Miniwatts Marketing Group, 2013. URL: http://www.internetworldstats.com.

[Mitsa 2009] T. Mitsa. Temporal Data Mining. Chapman & Hall CRC datamining & knowledge discovery series, 2009.

[Oliner and Stearley 2007] Adam Oliner and Jon Stearley. "What supercomputers say: a study of five system logs". In: *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pages 575–584.

[Perl] Official Perl Website. URL: http://www.perl.org/.

[R] R Project Website. R Project. URL: http://www.r-project.org/.

[Rails] Official Ruby on Rails Guide. URL: http://guides.rubyonrails.org/.

[Rathfelder et al. 2012] C. Rathfelder et al. "Workload-aware system monitoring using performance predictions applied to a large-scale e-mail system". In: *Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), Joint Working IEEE/IFIP*. 2012, pages 31–40.

[Rezagholi 2004] M. Rezagholi. Prozess- und Technologiemanagement in der Softwareentwicklung - Ein Metrik basierter Ansatz zur Bewertung von Prozessen und Technologien. Oldenbourg Wissenschaftsverlag GmbH, 2004.

[Rijsbergen 1979] C. J. Van Rijsbergen. Information Retrieval. 2nd. Butterworth-Heinemann, 1979.

[Salfner et al. 2010] Felix Salfner, Maren Lenk, and Miroslaw Malek. "A survey of online failure prediction methods". In: *ACM Computing Surveys (CSUR) Volume 42 Issue 3*. ACM New York, NY, USA, Mar. 2010.

[Sharma et al. 2003] Abhishek B. Sharma, Leana Glubchik, and Reamesh Govindan. "Sensor faults: detecion methods and prevalence in real-world datasets". In: *ACM Transactions on Sensor Networks Volume 6*. 2003, 23:1–23:39.

[Shasha and Zhu 2004] Dennis Elliot Shasha and Yunyue Zhu. High Performance Discovery in Time Series: Techniques and Case Studies. Springer, 2004.

[Shumway 2011] Robert Shumway and David Stoffer. Time Series Analysis and Its Applications 3rd Edition. Springer, 2011.

[Steinwart et al. 2005] Ingo Steinwart, Don Hush, and Clint Scovel. "A classification framework for anomaly detection". In: *Journal of Machine Learning Research Volume 6*. 2005, pages 211–232.

[van Hoorn et al. 2009] Andre van Hoorn et al. Continuous Monitoring of Software Services: Design and Application of the Kieker Framework. 2009.

[van Hoorn et al. 2012] Andre van Hoorn, Jan Waller, and Wilhelm Hasselbring. "Kieker: a framework for application performance monitoring and dynamic software analysis". In: *3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE'12)*. Apr. 2012, pages 247–248.

[Verbesselt et al. 2010] Jan Verbesselt et al. "Phenological change detection while accounting for abrupt and gradual trends in satellite image series". In: *Remote Sensing of Environment, Volume 114, Issue 12*. 2010, pages 2970–2980.

[Xing AG 2009] Forsa-Studie zeigt: Besser verdienende Führungskräfte sind in beruflichen Netzwerken wie XING deutlich aktiver. 2009. URL: `http://corporate.xing.com/index.php?id=108\&L=0\&tx\_ttnews[tt\_news]=774`.

[Xing AG 2012] Annual Report 3rd Quarter 2012. Xing AG, 2012. URL: http://corporate.xing.com/fileadmin/image_archive/XING_AG_ergebnisse_Q3_2012.pdf.

[Xing AG 2013a] Quarterly Report II/2013. Xing AG, 2013. URL: https://corporate.xing.com/fileadmin/IR/XING_Q2_2013_E.pdf.

[Xing AG 2013b] Xing Corporate Pages. Xing AG, 2013. URL: http://corporate.xing.com/no_cache/deutsch/unternehmen/xing-ag/.

[Xu et al. 2009] Wei Xu et al. "Online system problem detection by mining patterns of console logs". In: *Ninth IEEE International Conference on Data Mining*. 2009, pages 588–597.

[Yao 2010] Yuan Yao. "Online anomaly detection for sensor systems: a simple and efficient approach". In: *Perform. Eval.* Elsevier Science Publishers B. V. Amsterdam, The Netherlands, Nov. 2010, pages 1059–1075.

[ZeroMQ] Official ZeroMQ Website. URL: http://www.zeromq.org/.

[Zhang et al. 1998] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. "Forecasting with artificial neural networks: the state of the art". In: *International Journal of Forecasting Volume 14 Issue 1*. Mar. 1998, pages 35–62.

[Zou and Yang 2004] Hui Zou and Yuhong Yang. "Combining time series models for forecasting". In: *International Journal of Forecasting Volume 20 Issue 1*. 2004, pages 69–84.