QoS Prediction for Web Service Compositions Using Kernel-Based Quantile Estimation with Online Adaptation of the Constant Offset

Dries Geebelen[a,*], Kristof Geebelen[b,*], Eddy Truyen[b], Sam Michiels[b], Joos Vandewalle[a], Johan A.K. Suykens[a], Wouter Joosen[b]

[a] *ESAT-SCD-SISTA, Department of Electrical Engineering, K.U.Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium. Email address: dries.geebelen,joos.vandewalle,johan.suykens@esat.kuleuven.be*

[b] *IBBT-DistriNet, Departement of Computer Science, K.U.Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium. Email address: kristof.geebelen,eddy.truyen,sam.michiels,wouter.joosen@cs.kuleuven.be*

Abstract

This paper proposes a technique for predicting whether the Quality of Service (QoS) of a service composition execution will be compliant with a service level agreement (SLA) between a customer and the service (composition) provider. We assume time dependent QoS attributes of participating services and use time series prediction to obtain an accurate QoS estimate. This paper makes three contributions. First, to take into account dependencies between different services in a service composition, we propose a simulation technique based on Petri nets to generate composite time series using monitored QoS data of its elementary services. Second, we propose a *kernel*-based quantile estimator with *online adaptation of the constant offset*. The kernels allow the modeling of non-linearities of QoS attributes with respect to the input variables. The online adaption guarantees that under certain assumptions the number of times the predicted value is worse than the actual value converges to the agreed quantile value. Third, we introduce two performance indicators for comparing different QoS prediction algorithms. Our validation in the context of two case studies shows that the proposed algorithms outperform existing approaches by drastically reducing the violation frequency of the SLA while maximizing the usage of the candidate (composite) service.

Keywords: QoS, SLA, Service Composition, Time Series Prediction, Kernels, Online Learning, Quantile Estimation, WS-BPEL.

1. Introduction

Workflow languages focus on combining web services into aggregate services that satisfy the needs of clients. The Web Services Business Process Execution Language (WS-BPEL) [1]has profiled itself as the de-facto industry standard for orchestrating web services. A WS-BPEL process consists of a collection of related, structured activities or tasks that produce a specific service by combining services provided by multiple business partners. Tasks can be delegated to globally available software services and may require

*Both authors contributed equally to this work.

human interaction. For example, an integrated travel planning web service can be created by composing services for hotel booking, airline booking, payment, etc. With the rapidly growing number of such available services, service compositions are evolving from static processes and relatively simple service compositions, changing relative slowly over time, to global service networks that are complex and highly dynamic and which are actually evolving during execution to meet previously unknown requirements [15].

In a global service market, an important aspect of dynamic service composition is that service-based applications need the capability to change dynamically the participating services in order to satisfy customers' demand. Automatic service composition techniques that identify and invoke suitable services according to a service level agreement (SLA) require accurate QoS predictions prior to the execution of a given service composition. Candidate services can then be rejected or accepted to address a customers' request depending on wether these predictions indicate if the service execution will be done according to the SLA. Two types of errors can occur in this scenario: a type I error occurs when the service is rejected and would have satisfied the SLA constraints, a type II error occurs when the service is accepted and will violate the constraints. In the context of this paper, an SLA states that an accepted service should satisfy an SLA constraint with a probability greater than τ. Suppose f_τ is a function that outputs the predicted value for a given input and f_τ belongs to a certain hypothesis space $\mathcal{H}$, then the problem we will try to optimize becomes

$$\min_{f_\tau \in \mathcal{H}} Pr(\text{Type I error}|f_\tau)(1-\tau) + Pr(\text{Type II error}|f_\tau)\tau \tag{1a}$$

$$\text{such that } Pr(\text{Type II error}|\text{accepted}, f_\tau) \leq 1-\tau \tag{1b}$$

where Pr expresses a probability and 'accepted' means the service was accepted to execute the task. Constraint (1b) states that the frequency of accepting a service that violates the SLA constraint should not exceed $1-\tau$. Given (1b) holds, we have chosen the 'cost' we want to minimize in (1a) to be linear in the number of type I and type II errors: an error of type I has cost $1-\tau$ and an error of type II has cost τ. We will show in this paper that to minimize (1) one needs to know the conditional τ-quantile value. Problem (1) can also be reformulated as an optimization problem in the rejection rate ($Pr(\text{rejected})$) and the violation rate ($Pr(\text{Type II error}|\text{accepted})$):

$$\min_{f_\tau \in \mathcal{H}} Pr(\text{Type II error}|\text{accepted}, f_\tau) + Pr(\text{rejected}|f_\tau)((1-\tau) - Pr(\text{Type II error}|\text{accepted}, f_\tau))$$
$$\text{such that } (1-\tau) - Pr(\text{Type II error}|\text{accepted}, f_\tau) \geq 0. \tag{2}$$

where 'rejected' means the service was rejected to execute the task. In (2) we can see that the cost we want to minimize decreases for a decreasing rejection rate and a decreasing violation rate.

[1]Web Services Business Process Execution Language Version 2.0, April 2007, OASIS Technical Committee, `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html`

A challenge for a service composition provider is to solve this optimization problem for a composite service, given the QoS measures of the individual services. Considerable work has already been done on calculating QoS values of a composite service based on the QoS values of its constituents. As explained in [17], some existing approaches use *hard* composition rules such as addition, maximum or conjunction that combine predictions on the individual services to estimate composite quality measures [3, 4, 26]. In case one is interested in quantile values, hard contract rules can be overly pessimistic because they do not take the probability distribution of the individual services into account. This problem can be solved by estimating probability distributions for the QoS values of the elementary services which are combined according to *soft* composition rules [8, 17]. Soft contract rules, on the other hand, assume that QoS values of the different elementary services are independently distributed. In practice, quality attributes, such as response time, of different automated services can be dependent due to several reasons: both services run on the same server; during working hours certain services are executed more often than during the night; users choose the fastest service out of a group of services which causes the services to become equally fast; etc. Non-automatic service compositions that require human interaction to execute a task can also induce high correlations in QoS attributes during weekends when less people are available to execute a task. In these kind of scenarios, existing techniques using *soft* composition rules give distorted results.

This paper has the following contributions. We introduce two performance indicators to quantify our results: the first expresses, given certain assumptions, the cost we want to minimize in (1a) and the second expresses, given certain assumptions, the likelihood (1b) holds. We tackle the optimization problem discussed above by proposing a prediction algorithm based on kernel-based quantile estimation with online adaptation of the constant offset to predict if a service execution will be compliant with the SLA. This techniques is applicable on elementary as well as composite services. For the latter, our solution consists of two steps. First, simulation is done to generate a time series of QoS data for the composition based on the QoS data of the individual services. Second, we use our prediction algorithm on the resulting time series to predict if the composite service would be executed according to the SLA. Our approach is unique in the sense it has the following properties:

- It tries to minimize in a regularized manner and under certain assumptions the occurrence of type I and type II errors or, equivalently, the rejection rate and the violation rate.
- It can guarantee that, under certain assumptions, the number of times the predicted value is worse than the actual value converges to the agreed quantile value for the number of data points going to infinity. Moreover, it can guarantee that the second performance index will converge to 0.5.
- It allows the modeling of non-linear dependencies with respect to the input vectors. These input vectors can be chosen freely. They can contain the time, f.e. within a week, at which the service is invoked, past response times, etc. These input vectors can also be used to model seasonality.

- It does not assume that the probability distributions of a QoS-attribute of the composite's individual services are independent.

The remainder of this paper is organized as follows: Section 2 positions the paper in a broader context and elaborates on related work. Section 3 clarifies the problem statement and motivates our approach. Section 4 provides an overview of the QoS model we use for this work. We propose simulation technique based on Petri nets for calculating the QoS attributes of composite services, given QoS attributes of its elementary services in Section 5. Section 6 explains the performance indicators and describes the underlying prediction mechanism that is used to predict violations of the SLA between customer and service provider. An experimental evaluation of our approach and comparison with existing work is documented in Section 7. Finally, Section 8 concludes this paper and discusses future work.

2. Applicability & Related Work

In this section, we situate our paper in a broader context and elaborate on some challenges that we think are important to tackle the dynamic service selection problem for next-generation service-based systems. We also give a brief overview of related work that addresses each challenge.

Firstly, the automation of non-functional based service selection is a refinement of functional based service selection and requires that services need to be described in a way that can be 'understood' by computers. For example, automation of a travel planning service that needs to select a hotel service with certain quality requirements must be able to identify which services are feasible to fulfill the requirement of booking a room on a specific location. Semantic web services can provide a solution to this problem. Web service descriptions are enhanced with annotations of ontological concepts. Semantic matching can then be used to find an appropriate service to handle a specific task. Some popular standards that allow semantic annotation of web services are WSDL-S [2], SAWSDL [3], OWL-S [4], WSMO [5] and SWSF [6].

Secondly, there is a need to find an assignment of services to workflow tasks which maximizes a customer related utility function. The challenge is to find an optimal composition compliant with the SLA given the accurate QoS estimates of available services and an algorithm to calculate the QoS of the composition. First, we discuss related work that tackles the composition problem assuming fixed QoS attributes for the elementary services. Next, we elaborate on research that takes into account the fact that in business environments QoS attributes rarely remain unchanged over the lifetime of a web process and focus on the stochastic service composition problem.

[2] http://www.w3.org/Submission/WSDL-S/
[3] http://www.w3.org/TR/sawsdl/
[4] http://www.w3.org/Submission/OWL-S/
[5] http://www.w3.org/Submission/WSMO/
[6] http://www.w3.org/Submission/SWSF/

Cardoso's PhD. thesis [4] is a seminal work that proposes a framework that uses Stochastic Workflow Reduction to arrive at QoS estimates for the overall workflow, provided the QoS values for all tasks in the workflow are known. Although Cardoso et al. mentioned the possibility of deriving distribution functions for QoS of workflow tasks, the proposed reduction rules were applied to compute only fixed QoS values. Canfora et al. [3] apply a similar model with minor adaptations. Their middleware uses Genetic Algorithms for deriving optimal QoS compositions. Zeng et al. [26] also present a QoS-aware middleware for quality-driven web service compositions. In this work, the authors propose state charts and aggregation functions to represent the execution plans and execution paths. Two service selection approaches for constructing composite services have been proposed: local optimization and global planning. Their study shows that global planning is better than local optimization. A practical approach is taken by Mukherjee [13]. They propose a model for estimating three key QoS parameters - Response Time, Cost and Reliability - of an executable BPEL process from the QoS information of its partner services and certain control flow parameters.

Harney [7] present a composition solution that intelligently adapts workflow processes to changes in quality parameters of service providers. Changes are introduced by means of expiration times, i.e. service providers provide their current reliability rates and duration of time for which the current reliability rates are guaranteed to remain unchanged. Wiesemann et al. [25] formulate the service composition problem as a multi-objective stochastic program which simultaneously optimizes QoS parameters which are modeled as decision-dependent random variables. Their model minimizes the average value-at-risk (AVaR) of the workflow duration and costs while imposing constraints on the workflow availability and reliability.

Thirdly, an important challenge is to predict accurate expected values for quality measures prior to the execution of a given service composition. This prediction is not sufficient but necessary to be able to minimize the violation chance of a service level agreement (SLA) between customer and service provider in a volatile environment where QoS attributes of web services change over time. In this paper, we focus on this challenge that is restricted to predicting the quality properties of the individual services and the overall workflow. Related work is done by Rosario et al. [17]. They propose QoS estimation based on soft contracts. Soft contracts are characterized through probability distributions for QoS parameters. To yield a global contract for the composition, they use a tool called TOrQuE to unfold a composition and estimate its response time using Monte Carlo simulation. In contrast to our approach, their simulation technique assumes that the probability distribution of QoS values of the elementary services are independently distributed. As discussed in this work, this assumption is often violated in practice with as consequence that their approach leads to overoptimistic or overpessimistic results. We also compare our prediction technique with theirs in Section 7.2. Hwang et. al. [8] also approaches the service composition as a stochastic problem. Each QoS parameter for a web service is represented as a discrete random variable with a probability mass function. They propose a probabilistic framework to derive a QoS measure of a composite service from those of its constituent services and explore algorithms for computing the probability distribution functions of the QoS of the service composition. Again, their theoretical rules for composing the probability mass function are

based on the assumption that QoS values of each constituent web service of a composition construct are independent of those of the others. Shao et al. [19] proposes predictive methods based on making similarity mining and prediction from consumer experiences. Consumers that have similar historical QoS experiences on some services will likely have similar experiences on other services. They show that predicting QoS using their collaborative filtering based approach performs much better than average prediction.

3. Motivation

3.1. Running Example

This section presents a case study situated in the health care environment. The case study consists of a composite service (workflow), initiated by the government, that realizes a mammography screening program in order to reduce breast cancer mortality. The workflow is illustrated in Figure 1.

The first task of the workflow consists of sending out invitations to all women that qualify for the program. A radiologist will take images needed for screening and uploads them to the system (task 2). Next, the images need to be analyzed by specialized screening centers. There are always two independent readings, represented by tasks 3 and 4. These readings can be performed in parallel. In a next step, the two results of the readings are compared. When the results are identical, it is unlikely that the two physicians made the same mistake. Therefore it can be safely assumed that results are correct and the workflow can proceed with task 5. However, when the results are different, a concluding reading is performed (task 4'). Once the results of the screening of a particular screening subject are formulated, a report is generated (task 5) and a report is sent to the screening subject and her general practitioner in task 7. In parallel, different parties are billed (task 6).

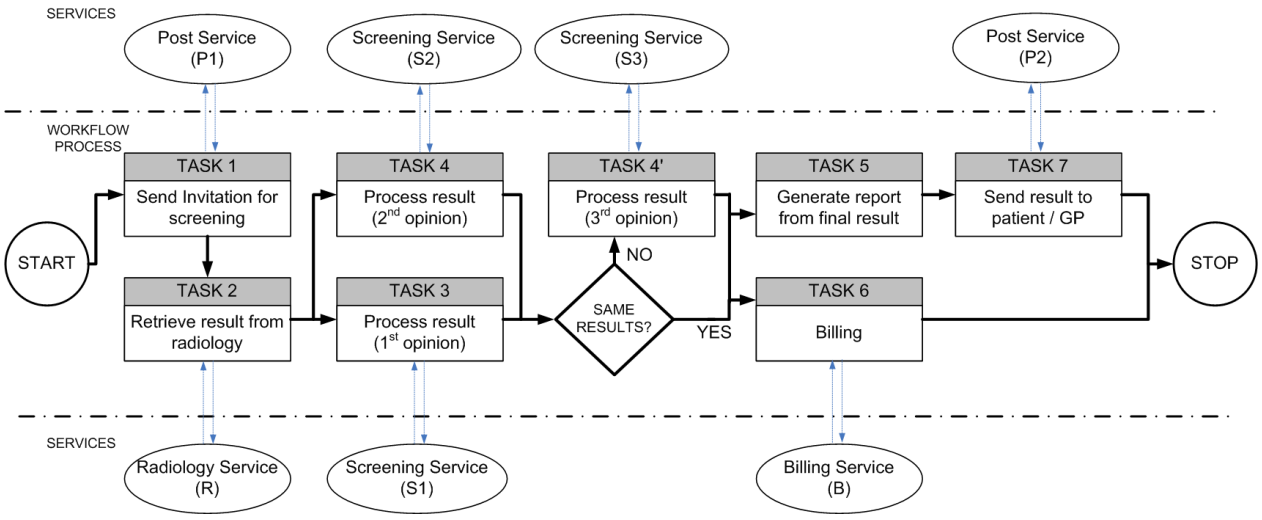


Figure 1: Example E-Health workflow process

Suppose the government, who finances this initiative, wants some quality guarantees and specifies a service level agreement with the company (service provider), that is responsible for executing the workflow. In this agreement, the company specifies that in $x\%$ of the cases the duration between task 1 and task 7 will take no longer than y working days. For simplicity, we consider only three tasks to explain the basic concepts in the rest of this section: the parallel execution of task 3 and 4 followed by a sequential execution of task 7. An example scenario is where the SLA states that in 99% of the cases the duration of these three tasks will take no longer than 3 working days. We will now show how a time dependent prediction algorithm in combination with quantile estimation can improve SLA compliance estimation of elementary and composite services.

3.2. Quantile vs. Average SLA Compliance Estimation

Services	Monitored RT (Prob. Density)					99% Q-Value	Average Value
	0-1	1-2	2-3	3-4	4-5		
CS 1	0%	1%	98%	1%	0%	3	2.5
CS 2	15%	75%	6%	3%	1%	4	1.5

Table 1: Probability density, average values and 99%-quantile values of RTs for 2 composite services CS1 & CS2.

The algorithms we propose are based on quantile estimation. We explain its benefit by means of a simple example. Suppose two different service selections on the e-health workflow leads to two composite services CS1 & CS2. The response time of these services have, at a certain time, probability densities for RTs of 0 to 5 days as presented in Table 1. We can use these values to make a quality estimate for an SLA. For example, which service would be the best candidate to have 99% certainty that its response time will not exceed 3 days? Using the average RTs of 2.5 days for CS1 and 1.5 days for CS2, it seems that CS2 is more reliable than CS1 and thus the best choice. However, using 99% quantile values of 3 days for CS1 and 4 days for CS2, we rightly conclude the opposite: CS1 has a higher probability that it will not exceed the threshold of 3 days since it is less volatile.

3.3. Time Dependent vs. Time Independent QoS Prediction

Again, consider a scenario where one has to estimate which of two composite services, as shown in Table 2, is the best candidate to comply with an SLA stating that the total duration of the composite service will take no longer than 3 working days. If we expand the pattern present in $t1$, $t2$, $t3$ and $t4$ to $t5$ and $t6$ as shown in Table 2, then at $t5$ 'CS A' complies with the SLA while 'CS B' violates the SLA and at $t6$ the

Services	Monitored RT (days)				Real RT (days)	
	t1	t2	t3	t4	t5	t6
CS A	2	5	2	5	**2**	**5**
CS B	4	2	4	2	**4**	**2**

Table 2: Past (monitored) and future (to be predicted) RTs for two composite services.

opposite happens. The estimation for this scenario, as for many real-life scenario's, is thus time dependent. Possible causes for varying quality of service attributes are temporary over -or underload, infrastructure failures, seasonality due to fixed working hours of non-automatic services, etc. The algorithms we propose to predict time varying QoS attributes are discussed in Section 6.

3.4. QoS Composition of Elementary Services

t	1	2	3	4	5	6	7	8	9	10
SS1	**1**	**3**	1	3	1	3	1	3	1	3
SS2	**1**	**1**	1	1	1	1	1	1	1	1
P	2	**1**	2	1	**2**	1	2	1	2	1
CS A	**2**	**5**	2	5	2	5	2	5	2	/

Table 3: Monitored response times (RTs) for the screening, post services and their composite service.

The estimation of QoS attributes can be done using monitored QoS values of elementary services. However, for a composite service provider, it would be interesting to have a solution that predicts QoS values of a composite service based on the QoS values of its constituents. Such a solution can for example be used to optimize the service selection process in correspondence with an SLA agreement. Suppose the composite service CS consists of two screening services (SS1 and SS2) that are executed in parallel and a post service (P) that is executed afterwards. The response time of these services vary in time. There are two possible strategies for making predictions of the composite service. The first strategy is to make predictions for the individual services and combine these predictions. The second strategy, which we will use, is to simulate a complete time series for the composite service, as it was executed several times in the past, and apply the prediction algorithm to this composite time series. The first strategy has the following disadvantages compared to the second strategy:

- Quantile values of QoS attributes of the individual service are not sufficient to calculate a realistic global quantile value for the composite service. One needs to predict the probability density for the QoS attributes of each individual service.

- The time at which an individual service will be executed depends on the response time of preceding services. To make accurate predictions for a service, one has to take this uncertainty of the start time into account.

- The quantile values depend not only on the probability density of the QoS values of the individual services but on the dependencies between these QoS values as well. These dependencies need to be modeled explicitly. The second strategy implicitly takes them into account.

On the example shown in Table 3 the simulated response time of the composite service is generated as follows according to the second strategy: at time 1 the parallel execution of SS1 and SS2 takes 2 time steps ($\max(1,1)$), this means P is executed at time 2 on which it has a response time of 1. The total response time of the composite service becomes 2 ($\max(1,1)+1$) at time 1. How to calculate the QoS values of a composite service based on the monitored values of the elementary services is explained in Section 5.

3.5. Quality of Web Service Data

A current problem for experiments regarding QoS of real web services is the lack of available datasets. For our analyses, we found no usable time series on Quality of Web Service attributes. Service providers usually only publish average values for their services. The QWS dataset [7] of Al-Masri et al. [1] includes measurements of 9 QoS attributes for 2500 real web services. Each service was tested over a ten-minute period for three consecutive days. However, only the average QoS values are publicly available. Another public dataset is WS-DREAM [8] which offers real invocation info on 100 web services by using 150 distributed computer nodes located all over the world. The dataset contains data on consecutive invocations of the services but is limited to 100 time series datapoints per service, which is not sufficient for our experiments. There is also no labeling on the time span of the different invocations of a service. Both datasets are restricted to short-running automated web services.

To cope with the data problem, we have collected real time series data for short-running online services ourselves. We used Web Inject [9], a free client-side monitoring tool for automated testing of web applications and web services. The tool allows to send soap requests to web services to analyze their response time and fault counts. We monitored a set of 8 popular online web services over a two-minute period for 7 consecutive days. Figure 2 illustrates the response time of a web service that allows a client to retrieve information on movies and theaters in the US. We can observe that a QoS attribute like response time can be very dynamic

[7] http: //www.uoguelph.ca/~qmahmoud/qws/index.html
[8] http://www.wsdream.net:8080/wsdream/
[9] http://www.webinject.org/

in time. Therefore, we believe that algorithms predicting QoS variations in time can contribute to existing literature on QoS-based web service composition.

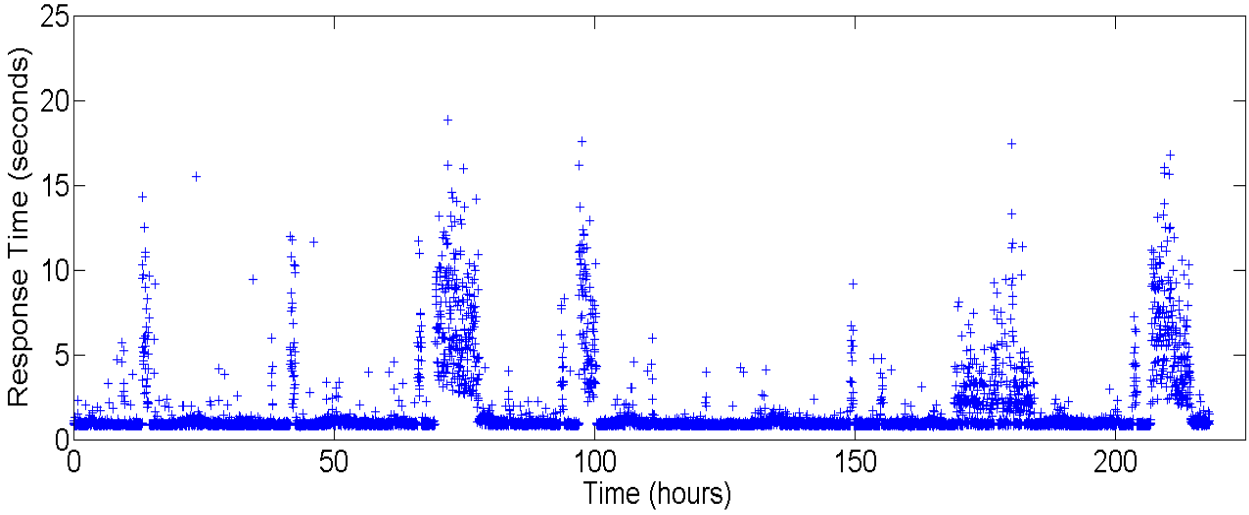


Figure 2: Time-varying RTs of online web service

We believe that our approach can also contribute in the area of long-running workflow processes where human intervention might be required to fulfill a task. An example of such process is the mammography screening workflow described in Section 3. Also airline industries define business processes where a task consists of bringing passengers luggage to an airplane, technical checkup of a plane, etc. Precise predictions are then crucial to avoid delaying take-offs. For this kind a services, it is even more difficult to find real datasets. We evaluate them by means of simulated data in Section 7.2.

4. QoS Considerations

4.1. QoS for Elementary Service

In the domain of web services, QoS parameters can be used to determine non-functional properties of the service. QoS attributes can be divided into quantitative and qualitative attributes. Examples of the latter are security and privacy. Popular quantitative attributes are response time, throughput, reputation, reliability, availability, and cost:

- Response Time (RT): the time taken to send a request and receive a response (expressed in milliseconds). The response time is the sum of the processing time and the transmission time. For short running processes they are usually of the same order. For long running processes that can take hours, days or even weeks to complete, the transmission time is usually negligible.

- Throughput (TP): the maximum requests that can be handled at a given unit in time (expressed in requests/minute).
- Reputation (RP): the reputation of a service is a measure of its trustworthiness (expressed as scalar with higher value being better). The value is defined as the average ranking given to the service by end users.
- Reliability (RL): the probability that a task is satisfactorily fulfilled (expressed as a percentage). The reliability can be calculated from past data by dividing the number of successful executions by the total number of executions.
- Availability (A): the probability that a web service is available (expressed in available time/total time). It is computed by dividing the total amount of time in which a service is available through the total monitoring time. In the scope of this work, we define an available service as a service that is able to responds within a predefined time interval.
- Cost (C): the cost that a service requester has to pay for invoking a specific operation of a service (expressed in cents/request). Other pricing schemes are sometimes used such as membership fee or monthly fee.

Since the focus of this paper is on the prediction of quality of service attributes with a volatile nature, we do not consider static attributes like fixed costs such as membership fees. Also reputation is an attribute that gives a general impression about users opinions of a service, and is not meant to frequently change in time. System-level QoS attributes, such as throughput, often largely depend on hardware and computing power of the underlying infrastructure of the composite service and need to be evaluated over multiple instances. In this work, we focus on instance-level QoS attributes of composite services that directly relate to the QoS values of its constituent web services and moreover are non-stationary. Interesting attributes for our approach are response time, reliability, availability and cost as pay-per-service. For the sake of simplicity we limit the QoS prediction algorithms in Section 6 and their evaluation in Section 7 to the response time attribute.

4.2. QoS for Composite Services

The QoS of a service composition is calculated based on the QoS values of its constituents. In contrast to the measurement of QoS for elementary services, composite services consist of different activities such as sequences, if-conditions, loops and parallel invocations. We need to take into account these different composition patterns to calculate the QoS of a composite service. Example QoS computations are summarized in Table 4. WS-BPEL elements relevant to QoS computation are simple elements as receive, reply, invoke, assign, throw, wait and complex elements like sequence, flow, if, while and foreach. Similar to Kiepuszewski

QoS Attribute	Composition Patterns			
	Sequence	**Parallel**	**Switch**	**Loop**
Response Time	$\sum_{i=1}^{m} RT_i$	$\max(RT_i)$	$\sum_{i=1}^{m} p_j.RT_i$	$RT.k$
Throughput	$\min(TP_i)$	$\min(TP_i)$	$\sum_{i=1}^{m} p_j.TP_i$	TP
Reputation	$\sum_{i=1}^{m} \frac{RP_i}{m}$	$\sum_{i=1}^{m} \frac{RP_i}{m}$	$\sum_{i=1}^{m} p_j.RP_i$	RP
Reliability	$\prod_{i=1}^{m} RL_i$	$\prod_{i=1}^{m} RL_i$	$\sum_{i=1}^{m} p_j.RL_i$	RL^k
Availability	$\prod_{i=1}^{m} A_i$	$\prod_{i=1}^{m} A_i$	$\sum_{i=1}^{m} p_j.A_i$	A^k
Cost	$\sum_{i=1}^{m} C_i$	$\sum_{i=1}^{m} C_i$	$\sum_{i=1}^{m} p_j.C_i$	$C.k$

Table 4: QoS Computations for Composite Services

et al. [10], we define a structured model that consists of four constructs that allow for recursive construction of larger workflows:

- Sequence: multiple tasks that are sequentially executed.
- Parallel execution (and-split/and-join): multiple paths that are executed concurrently en merged synchronously.
- Exclusive choice (or-split/or-join): multiple possible paths, among which only one can be executed.
- Loop: a path that is repeatedly executed a fixed number of times c.

Various standards for service composition include more constructs in addition to the four basic constructs described above. Jaeger et al. [9] summarizes the workflow patterns that cover most control constructs proposed in existing standards and products. In this paper we limit ourselves to the four basic constructs that are able to cover the most important activities offered in WS-BPEL, a workflow language which has currently profiled itself as the de-facto industry standard for orchestrating web services. How the composite activities of WS-BPEL are mapped to the basic constructs will be explained in the next section. A Petri net graph and a Petri net execution time system (PNET-system) is used to reason over the workflow and estimate its total response time.

Besides the overhead generated by the service calls, the QoS of a composite service is influenced by events internal to the orchestration. Usually, the delay caused by internal events is negligible compared to that of the service calls. This is certainly the case for medium and long running processes, which are the main targets for our approach. For further analysis, we assume that the overall delay of the orchestration depends solely on the response times of the services it calls during execution. The inclusion of internal delays is a trivial extension.

5. Simulated QoS for Composite Services

In the introduction, we emphasized the limitations of existing works that use hard or soft composition rules to estimate composite QoS attributes. In this section we try to overcome these limitations by generating a simulated time series of the QoS attributes of the workflow as if the composition was executed several times in the past. The quantile values will be estimated using this simulated dataset as will be explained in Section 6.

5.1. Petri Net Graph

Various activity-based process models have been suggested for workflow systems. To explain the algorithm, we represent the workflow as a non-deterministic Petri net graph (PNG), which is a commonly used representation for workflow systems [18, 14]. Compared to a structured workflow model, expressing our QoS model as a formal model offers more expressive power and is equipped with strong analysis capabilities. There are two kinds of transitions in our Petri net: timed and immediate transitions. Timed transitions represent services and the firing delay of the transition corresponds to the response time of these services. Immediate transitions are needed to enable the representation of internal events of the composite service. A Petri net graph contains places as well. Places correspond to workflow states and allow the representation of conditional execution.

A generalization of the Petri net graph we use for our analysis of WS-BPEL, is a 8-tuple ($\mathcal{P}$, $\mathcal{T}^1$, $\mathcal{T}^2$, $\mathcal{T}$, $\mathcal{D}$, W^-, W^+, s_0), where

- $\mathcal{P} = \{p_1, p_2, ..., p_{n_1}\}$ is a finite set of places. The set contains exactly one starting place (p_1) and exactly one ending place (p_{n_1}).
- $\mathcal{T}^1 = \{t_1^1, t_2^1, ..., t_{n_2}^1\}$ is a finite set of immediate transitions. Immediate transitions have no firing delay.
- $\mathcal{T}^2 = \{t_1^2, t_2^2, ..., t_{n_3}^2\}$ is a finite set of timed transitions. Timed transitions have a firing delay.
- $\mathcal{T} = \{t_1, t_2, ..., t_{n_4}\}$ is a finite set of transitions. All transitions are immediate or timed transitions ($\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^2$) but cannot be both ($\mathcal{T}^1 \cap \mathcal{T}^2 = \emptyset$).
- $\mathcal{D} = \{d_1, d_2, ..., d_{n_2}\}$ is finite set of positive real functions representing the firing delay of the corresponding timed transitions with respect to the time. $d_i(s)$ is the firing delay of t_i^2 at time s.
- W^-, W^+ are the backward and forward incidence matrices, respectively. They contain boolean values. If $W^-(i,j) = 1$, then there is an arc going from p_i to t_j. If $W^+(i,j) = 1$, then there is an arc going from t_i to p_j. If $W^-(i,j)$ or $W^+(i,j)$ equals 0, then there is no corresponding arc. Timed transitions

have one incoming arcs and one outgoing arc. Places, except for the starting and ending place, can have multiple incoming arcs ('OR-join') or multiple outgoing arcs ('OR-split'). The starting place differs in the sense it has no incoming arcs and the ending place differs in the sense it has no outgoing arcs. Immediate transitions can have multiple incoming arcs ('AND-join') or multiple outgoing arcs ('AND-split').

- The initial marking of the Petri net is always one token present at the starting place. The initial time of the Petri net equals s_0. Time is continuous and can take any real value.

- An additional constraint on our Petri net is that each 'AND-split'-transition, 'OR-split'-place has to have exactly one corresponding 'AND-join'-transition, 'OR-join'-place respectively and vice versa. With corresponding we mean that all outgoing paths of the 'split'-node are disconnected until they reach the corresponding 'join'-node in which they *all* come together. The paths connecting the 'split'-node and corresponding 'join'-node are called the connecting paths.

The mapping of WS-BPEL activities, represented by their Business Process Modeling Notation (BPMN) [10], to their corresponding Petri net representation is shown in Figure 3. If we apply this mapping on the case study introduced in Section 3.1, we get the Petri net representation illustrated in Figure 4. The firing delays introduced by the timed transitions t_1^2 to t_7^2 correspond to the response times of the post 1, radiology, screening 1 to 3, report, billing and post 2 service respectively. Remark that we have truncated the Petri net graph by removing 'a place followed by an immediate transition with one incoming and one outgoing arc'. These constructs have no influence on further analysis and calculations.

Similar to Colored Petri nets, we add an extension to the elements of the net to deal with the other QoS attributes besides response time: A Petri net token is associated with 3 data values C, A and RL of type integer to hold the current aggregated cost, availability and reliability of a composition. The extension is a 4-tuple ($\mathcal{Q}$, $\mathcal{C}$, $\mathcal{A}$, $\mathcal{RL}$), where

- $\mathcal{Q} = \{q_1, q_2, ..., q_{n_3}\}$ is a finite set of positive real numbers representing the maximal allowed delay for each timed transition before it is considered as unavailable.

- $\mathcal{C} = \{c_1, c_2, ..., c_{n_3}\}$ is a finite set of positive real functions representing the cost of the corresponding timed transitions with respect to the time. $c_i(s)$ is the cost of t_i^2 at time s.

- $\mathcal{A} = \{a_1, a_2, ..., a_{n_3}\}$ is a finite set of functions where $a_i(s) \in \{0, 1\}$ represents the availability of the

[10] Business Process Modeling Notation (BPMN) Version 2.0, January 2011, OMG Specification, `http://bpmn.org/`

corresponding timed transitions with respect to the time. $a_i(s)$ is the availability of t_i^2 at time s. There is a direct relation between a_i and d_i: if $(d_i <= q_i)\{a_i = 1\}$ else $\{a_i = 0\}$

- $\mathcal{RL} = \{rl_1, rl_2, ..., rl_{n_3}\}$ is a finite set of functions where $rl_i \in \{0, 1\}$ represents the reliability of the corresponding timed transitions with respect to the time. $rl_i(s)$ is the reliability of t_i^2 at time s.

5.2. Petri Net Execution Semantics

The execution of the Petri net graph is done by passing tokens from the initial marking to the end marking. These markings correspond to a token present at start place and end place respectively. An execution of the Petri net graph simulates the execution of the workflow. The execution (response) time of the workflow at time s is the time elapsed between a token present at the starting place at time s and a token reaching the ending place in the same execution cycle. During execution of the Petri net, time increases in a continuous way, starting from the initial value s_0. For each time, the following rules are executed in a non-deterministic order until no more rules can be executed:

- Rule 1: If a token is present at the incoming place of a timed transition t_i^2, then that token is consumed and the fire delay counter of that transition is activated. The counter starts counting down from $d_i(s)$ where s is the current time. As time increases, the counter value decreases with an equal amount.
- Rule 2: If an active fire delay counter of a timed transition reaches zero, then a token is generated at the outgoing place and the fire delay counter is deactivated.
- Rule 3: If tokens are present at *all* incoming places of an immediate transition, then these tokens are consumed and new tokens are generated at all outgoing places.
- Rule 4: If a token reaches the ending place, then the execution stops. Time freezes and no more rules are executed. The execution time of a Petri net graph equals the time at which the execution stops minus the initial time.

Applied on our case study, the token starts at the start place at s_0. The only applicable rule is rule 1 where the fire delay counter of the timed transition t_1^2 is activated. When the counter reaches zero at time $s_0 + d_1$, a new token is generated according to rule 2 and arrives at P1. Analogue, the token reaches P2. At P2 rule 3 is executed and new tokens are generated at both outgoing places of t_1^1. The parallel execution is done according to rule 1 followed by 2 for each branch in a non-deterministic order. The parallel execution ends when both tokens are consumed and a new token is generated at the immediate transition t_2^1 according

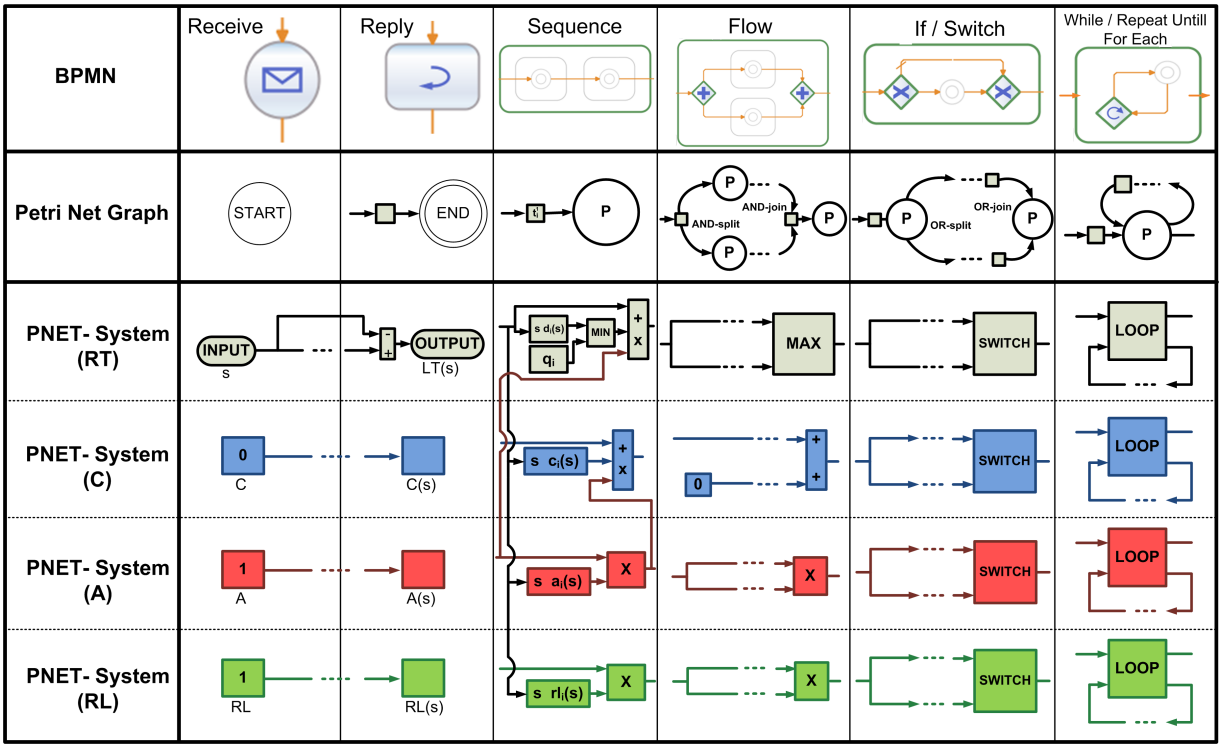


Figure 3: Mapping: Business Process Modeling Language (BPMN) - Petri net graph - PNET-system

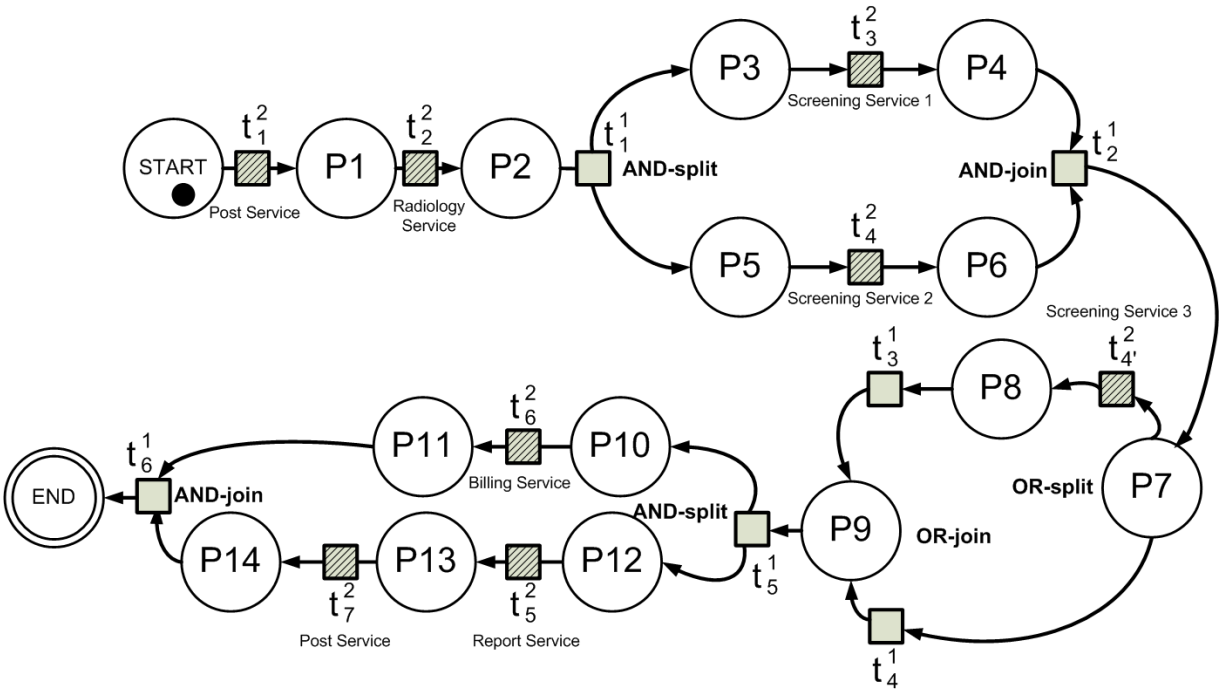


Figure 4: Petri net graph for e-health workflow

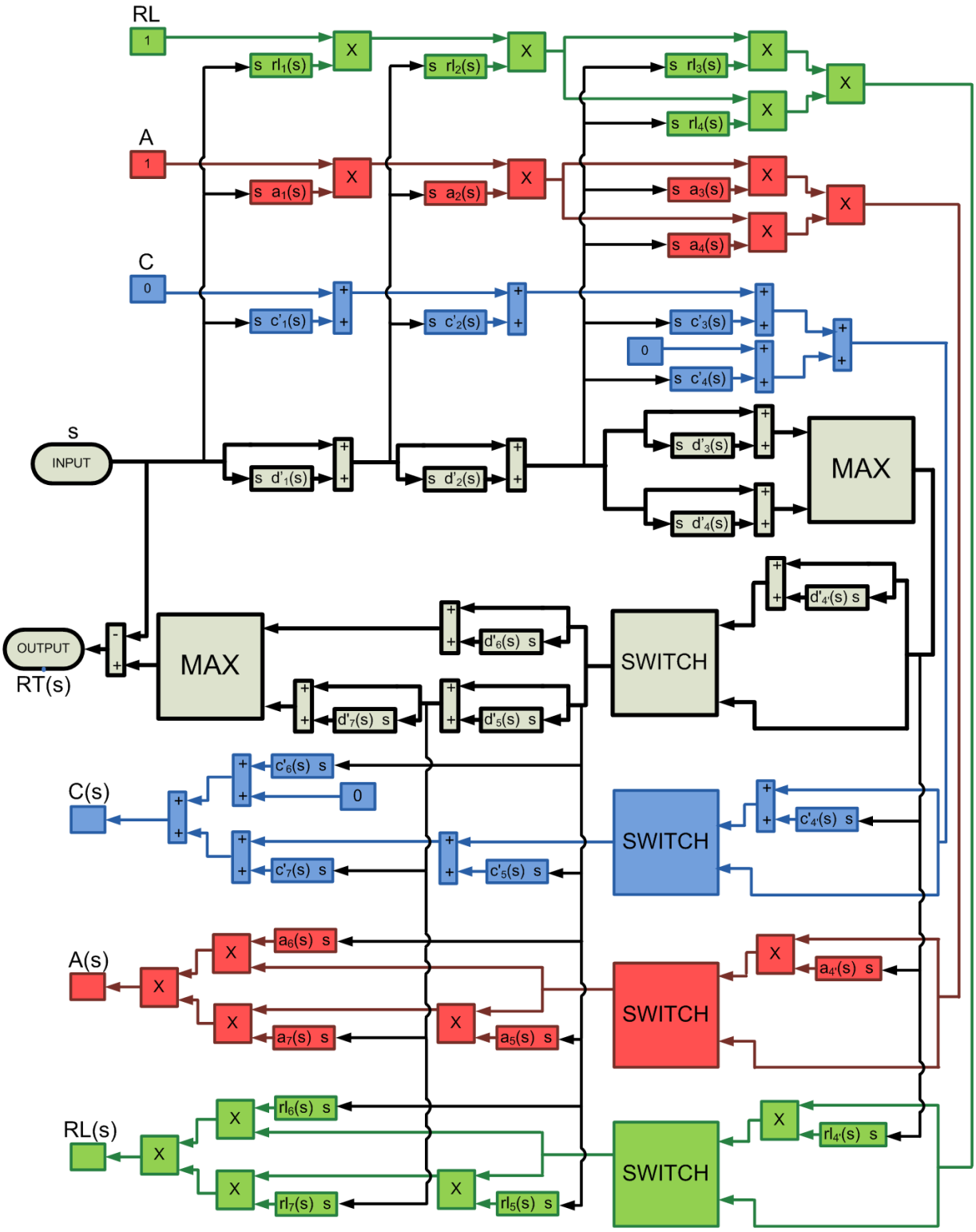


Figure 5: Petri net execution time system for e-health workflow

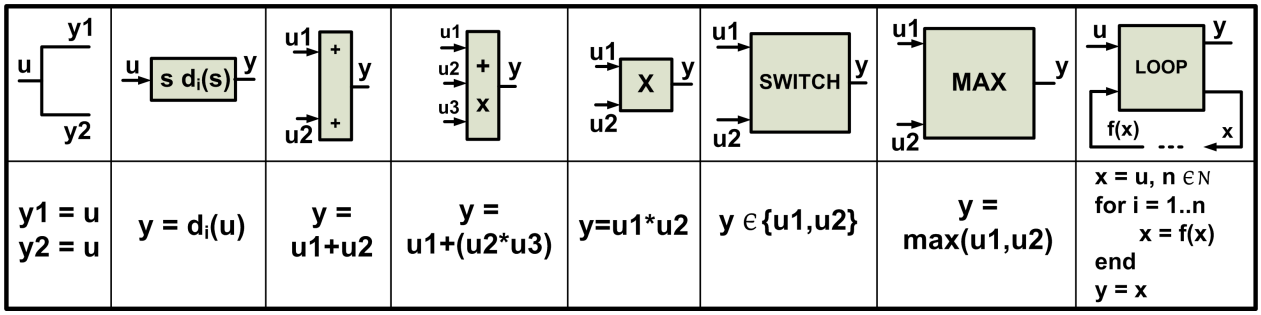


Figure 6: Overview of the PNET-system building blocks

to rule 3. When this token arrives at P7, both rule 1 and rule 3 can be executed corresponding to the upper and lower path respectively. Again, a rule is chosen non-deterministically. If rule 3 fires, a token is generated at the lower path and sent to the immediate transition. If rule 1 fires, a token is generated at the upper path, going to the timed transition $t^2_{4'}$. The next steps are similar to what we explained already. The execution ends after the 'AND-join', where rule 4 is applied and the token reaches its ending state. The execution time of the Petri net graph, where tokens are passed from start to end, corresponds to the simulated response time of the composite service.

To keep track of the cost, availability and reliability during the execution of the Petri net, we add the following rules as an extension to the existing ones:

- Rule 0: If a token is present at the start place, all associated data values are initialized as follows: $C \leftarrow 0, A \leftarrow 1, RL \leftarrow 1$.
- Rule 1': If the firing delay $d_i(s)$ of a timed transition t^2_i is above a predefined value q_i, we consider the timed transition as unavailable. The data values associated with the token are updated as follows: $A \leftarrow 0, C \leftarrow C, RL \leftarrow 0$ and the Petri net execution is halted. The Petri net execution time for the halted service then equals q_i. The response time now equals the time elapsed between a token present at the starting place and the time at which the execution is halted.
- Rule 2': If an active fire delay counter of a timed transition t^2_i reaches zero, then a token is generated at the outgoing place and the fire delay counter is deactivated. The data values associated with the token are updated as follows: $A \leftarrow 1, C \leftarrow C + c_i(s), RL \leftarrow RL \times rl_i(s)$.

Taking into account these extension rules for the case study, the data values C, A and RL contain the current aggregated QoS values for the cost, availability and reliability respectively.

5.3. Petri Net Execution Time System

Instead of simulating the above Petri net graph, we derive in this subsection a transformation of a Petri net graph into a Petri net execution time system (PNET-system). A PNET-system immediately outputs the time at which the execution of the corresponding Petri net graph stops, given the time at which the execution starts. The extended PNET-system also outputs the corresponding cost, availability and reliability of the Petri net execution. Because the Petri net is nondeterministic, the PNET-system is nondeterministic as well. An overview of the building blocks is illustrated in Figure 6.

Definition 1. *We define* $S_i^-, C_i^-, A_i^-, RL_i^-$ *and* $S_i^+, C_i^+, A_i^+, RL_i^+$ *as the values of the simulation time, cost, availability and reliability right before, respectively right after the execution of the timed transition* t_i^2*. The simulation time* s *represents the virtual time during which the Petri net is executed.*

The PNET-system is generated as follows out of a Petri net graph for the different QoS attributes (see also figure 3 for the relationship between their constructs):

Response Time:

- Instead of calculating the execution (response) time, we are, until now, calculating the time at which the execution stops. For that reason we subtract the initial time (s_{start}) from the time at which the execution stops: $RT(s) = s - s_{start}$.
- The time at which a token is generated at the outgoing place of a timed transition t_i^2 equals the time at which a token is consumed at the incoming place added with the firing delay $d_i(s)$. A timed transition in a PNG corresponds to an addition with the delay at that time in a PNET-system. In the extended Petri net execution, any delay above the predefined value q_i is not taken into account since the workflow is then considered unavailable. Delays on an unavailable path are also not taken into account to calculate the simulated response time. The simulation time after the timed transition t_i^2 is fired at time s is thus calculated as $S_i^+ = S_i^- + \min(d_i(s), q_i) * A_i^-$.
- An 'AND-split' means all paths are executed simultaneously. The time at which the 'AND-split' is fired is copied to all outgoing places. An 'AND-split' in a PNG corresponds to a 'fork' in a PNET-system. The time at which a token is fired at the 'AND-join'-transition equals the maximum of the times on which a token is generated at the incoming places. An 'AND-join'-transition in a PNG corresponds to a 'MAX'-building block in a PNET-system.
- An 'OR-split' with corresponding 'OR-join' means a token is send into one of the connecting paths. This is equivalent to sending tokens into all connecting paths, but all but one of the former connecting

paths are disconnected from the former 'OR-join'. An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.

- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

Availability & Reliability:

- The initial availability (reliability) is initialized to 1 (100%).
- An 'AND-split' in a PNG corresponds to a 'fork' in a PNET-system. An 'AND-join'-transition in a PNG is where two parallel paths meet. The resulting availability (reliability) is the product of availability (reliability) of both paths. An 'AND-join'-transition in a PNG corresponds to a multiplication-building block in a PNET-system.
- An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.
- The availability (reliability) at which a token is generated at the outgoing place of a timed transition equals the availability (reliability) at which a token is consumed at the incoming place multiplied with the availability (reliability) at the simulated execution time (starting time plus the time induced by the firing delays of the previous timed transition). Current simulated execution time is thus used as an input to retrieve the availability (reliability) at a specific simulated execution point.
- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

Cost:

- The initial cost is initialized to 0.
- The cost of a parallel execution equals the summation of the costs generated on all paths. To take into account the cost generated before the parallel execution, one path is connected with the previous execution path by a straight line. All other paths start counting form zero. An 'AND-join'-transition in a PNG corresponds to a summation-building block in a PNET-system. The result is a summation of all prior costs.
- An 'OR-split' in a PNG corresponds to a 'fork' in a PNET-system and an 'OR-join' in a PNG corresponds to a 'SWITCH'-building block in a PNET-system.

- The cost at which a token is generated at the outgoing place of a timed transition equals the cost at which a token is consumed at the incoming place plus the cost at the current simulated execution time. Also here the current simulated execution time is a necessary input to retrieve the cost. When a service is not available, the execution of the PNG stops and no further costs are made. In the PNET-system this is achieved by multiplying the cost generated by the timed transition with the aggregated availability of the system. The resulting cost after the timed transition t_i^2 that is fired at time s is thus calculated as $C_i^+ = C_i^- + c_i(s) * A_i^+$.

- A loop in a PNG corresponds to a 'LOOP'-building block in a PNET-system.

The extended PNET-system for the e-health workflow is illustrated in Figure 5. To keep the overview, we did not include all the building blocks as defined in Figure 6 but used the modified functions $d_i'(s) = \min(d_i(s), q_i) * A_i^-$ and $c_i'(s) = c_i(s) * A_i^+$ instead.

5.4. From Non-Deterministic to Deterministic

To make our approach practically usable, we need to find a way to simulate a nondeterministic system using a deterministic algorithm. To allow ex-ante estimation of QoS values, we need to make assumptions on how many times a loop will be executed and which path will be followed after a conditional execution. This is not always trivial since the number of loop executions or the value of a condition is often only known at run-time. Possible strategies for resolving the non-deterministic constraints by making deterministic assumptions are:

- Assume the worst-case scenario. In case of a loop, this means to use the maximum number of times it can be executed in practice. For a conditional execution, the worst-case path depends on the QoS attribute. The worst response time is on the path that takes the longest time to execute. This can be modeled by replacing the 'OR-split' and 'OR-join' by an 'AND-split' and 'AND-join' (parallel execution with synchronization) in the Petri net graph and replacing the 'SWITCH' by a 'MAX' building block in the PNET-system. The worst-case cost is on the most expensive path and for availability (reliability), it is the least available (reliable) path. In practice, one can simulate the QoS values for all paths and take the worst values for each attribute. For a more general approach to worst case execution analysis, we refer to specialized literature in this domain [16].

- Use a probabilistic model by assigning probabilities to the number of times a loop is executed or to each path that is implied by a condition. In practice, this strategy can be realized by doing the simulation for all paths considered and assign probabilities to the resulting QoS values. An important remark

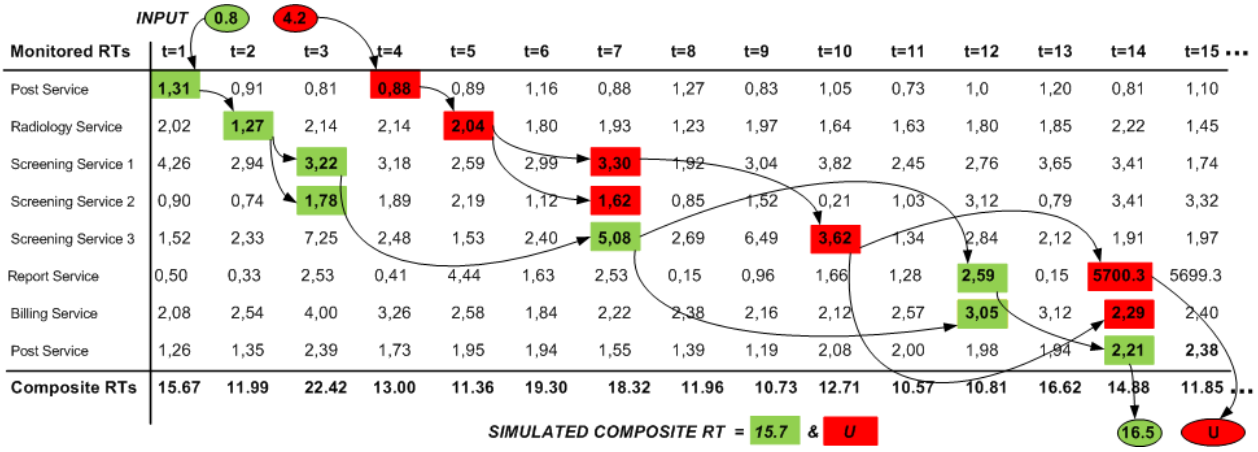


Figure 7: Example simulated RT calculation for the e-health workflow

here is that the resulting values cannot simply be added after multiplication with their corresponding probabilities if one wants to estimate quantile values.

5.5. Example Simulated QoS Calculation

In this section, we explain more in detail how we generate a composite time series from individual time series of the constituting services using the PNET-system. Suppose we have monitored the response times for several consequent time periods of the 8 services used in the mammography workflow as shown in Figure 7. We generate a virtual time series by simulating the execution of the composite service according to the PNET-system discussed in the previous subsection. The inputs are fixed time steps in the past. For example, if the workflow would be executed at time 0.8, we can see that the execution of the first service (post service) takes approximately 1.31 time units. This implies that the second service (radiology service) will be executed at time 2.11 (sum of 0.8 and 1.31). The closest monitored result is at t=2 with a response time of 1.27 units. The resulting execution time is 3.38 (sum of 2.11 and 1.27). At this point, the two screening services S1 and S2 will be executed in parallel with corresponding RTs of 3.22 and 1.78. Since this is a parallel execution where the workflow has to wait for the slowest service to finish, we will arrive at the next service at time 6,61 (maximum ending time of SS1 and SS2). The other calculations are similar. For the if condition, we have to make a deterministic assumption concerning the path that will be chosen. In practice, in most of the cases of a mammography screening the results of SS1 and SS2 will probably match, meaning that a third opinion is not necessary. This implies that the upper path of the conditional execution in Figure 5 rarely will be chosen. Nevertheless, the safest strategy is to calculate the worst case scenario. This means we take into account the path that generates the maximum response time of all the paths that could be implied by the if

Monitored QoS	q_i	**t**	**RT**	**C**	**A**	**RL**
Post Service	2.10	1	1.31	6.72	1	1
		4	0.88	5.77	1	1
Radiology Service	6.14	2	1.27	203.30	1	1
		5	2.04	235.33	1	1
Screening Service 1	6.32	3	3.22	110.37	1	1
		7	3.30	110.89	1	1
Screening Service 2	20.15	3	1.78	103.17	1	1
		7	1.62	102.62	1	1
Screening Service 3	15.95	7	5.08	125.81	1	1
		10	3.62	113.10	1	1
Report Service	49.35	12	2.59	6.71	1	1
		14	5700.3	8.57	0	0
Billing Service	4.82	12	3.05	9.30	1	1
		14	2.29	5.24	1	0
Post Service	5.15	14	2.21	9.88	1	1
		-	-	-	-	-
Composite QoS	$s = 0.8$		15.67	575.26	1	1
	$s = 4.2$		63.6	572.95	0	0

Table 5: Monitored QoS values for corresponding execution times.

condition. Worst case calculations of the response time of the composite service using input times (s_{start}) of 0.8 and 4.2 are:

$$RT_{0.8} = 0.8 + 1.31 + 1.27 + \max(3.22; 1.78) + \max(5.08; 0) + \max(2.59 + 2.21; 3.05) - 0.8 = 15.7$$

$$RT_{4.2} = 4.2 + 0.88 + 2.04 + \max(3.30; 1.62) + \max(3.62; 0) + \max(5700.3 + U; 2.29) - 4.2 = U$$

Remark that for the second calculation, the report service is unavailable and takes 5700.3 s to recover from failure. For services further in the execution chain, QoS values cannot be retrieved because no data is available that far in the future. An unavailable value is marked as 'Unknown (U)' and has the following properties: $0 \times U = 0; 1 \times U = U; \max(U, x) = U; \min(U, x) = U; U + x = U$. In our extended calculation below, we tackle this problem by taking into account the other QoS attributes and their interrelations.

To simulate the composite QoS values for cost, availability and reliability, we need their monitored values for all participating services at the times they are actually executed.Figure 5 shows some fictive QoS values we use to illustrate our calculations. When a service on the path of execution is not available, the aggregated availability and reliability will become 0 and the costs and response times of the remaining services will not be taken into account due to its multiplication with the aggregated availability in the PNET-system. Worst case calculation for input times of 0.8 and 4.2 are:

$$
\begin{aligned}
RT_{0.8} &= 0.8 + \min(1.31, 2.10) \times 1 + \min(1.27, 6.14) \times 1 + \max(\min(3.22, 6.32) \times 1; \min(1.78, 20.15) \times 1) \\
&\quad + \max(\min(5.08, 15.95) \times 1; 0) + \max(\min(2.59, 49.35) \times 1 + \min(2.21, 5.18) \times 1; \min(3.05, 4.82) \times 1) - 0.8 \\
&\quad = 15.7 \\
A_{0.8} &= 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1 \\
C_{0.8} &= 6.72 \times 1 + 203.30 \times 1 + 110.37 \times 1 + 103.17 \times 1 + 125.81 \times 1 + 6.71 \times 1 + 9.30 \times 1 + 9.88 \times 1 \\
&\quad = 575.26 \\
RL_{0.8} &= 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1
\end{aligned}
$$

$$
\begin{aligned}
RT_{4.2} &= 4.2 + \min(0.88, 2.10) \times 1 + \min(2.04, 6.14) \times 1 + \max(\min(3.30, 6.32) \times 1; \min(1.62, 20.15) \times 1) \\
&\quad + \max(\min(3.62, 15.95) \times 1; 0) + \max(\min(5700.3, 49.35) \times 1 + \min(U, 5.18) \times 0; \min(2.29, 4.82) \times 1) - 4.2 \\
&\quad = 63.6 \\
A_{4.2} &= 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 0 \times 1 \times 1 = 0 \\
C_{4.2} &= 5.77 \times 1 + 235.33 \times 1 + 110.89 \times 1 + 102.62 \times 1 + 113.10 \times 1 + 8.57 \times 0 + 5.24 \times 1 + U \times 0 \\
&\quad = 572.95 \\
RL_{4.2} &= 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 0 \times 0 \times U = 0
\end{aligned}
$$

Using this technique to calculate a time series for the composite service, we can apply the same prediction algorithm to estimate the QoS for elementary as composite services. An important advantage of this approach is that we do not have to assume that the QoS attributes of the constituting services of a composite service are independently distributed.

6. QoS Prediction

In this section we discuss methods to predict QoS attributes, more specifically we will focus on response times. We propose a kernel-based quantile estimator with online adaptation of the constant offset for the following reasons:

- Quantile estimation determines whether the likeliness that a certain constraint will be satisfied is larger than a predefined value. For this application, knowing the quantile value can be more interesting than knowing the average or median value.

- We chose a kernel-based method because using non-linear kernels allows the learning of non-linear dependencies. Kernel-based methods like SVM [23] and LS-SVM [21], have also shown to be successful for various applications such as optical character recognition and electricity load prediction.
- For batch learning, good training and validation set performance are no guarantee for good test set performance. It is, for example, possible that the dynamics of the system change in the test set or that certain events, not present in the training and validation set, happen in the test set. This can cause the number of violations to exceed the agreed value. The additional online adaptation of the constant offset we propose in this paper, makes sure the number of times the estimated response time exceeds the true response time converges to the agreed quantile value.

Section 6.1 gives a brief introduction to kernel-based regression. In Section 6.2 we discuss two performance indicators we want to minimize out-of-sample. A kernel-based bath-learning algorithm designed to minimize the first performance indicator is discussed in Section 6.3. Section 6.4 explains an online algorithm designed to minimize the second performance indicator. Finally Section 6.5 combines both of the previous algorithms such that a good performance is achieved on both indicators.

6.1. Kernel-based Regression

Regression is a common task in machine learning. Given a training set $\mathcal{S}$ containing input vectors $\{x_{t,tr}\}_{t=1}^{n_{tr}}$ along with corresponding output values $\{y_{t,tr}\}_{t=1}^{n_{tr}}$, the task is to find a function f that defines the relation between these input vectors and output values such that the errors on unseen data are as small as possible. The performance of the learned function f is evaluated on a testset $\mathcal{T}$ containing n_{test} datapoints. We assume that the function f belongs to a Reproducing Kernel Hilbert Space (RKHS) with k the corresponding kernel function. A popular kernel function is the radial basis function (RBF) kernel ($k(x, y) = \exp(-\|x-y\|_2^2/\sigma^2)$). Readers not familiar with support vector machines are recommended to read a tutorial, such as [20]. The loss function $l(f(x_t), y_t)$ represents the cost of each error. Commonly used loss functions within kernel-based methods are the ϵ-insensitivity loss function (3) [23] and the quadratic loss function (4):

$$l_\epsilon(y_t - f(x_t)) = \begin{cases} 0, & \text{if } |y_t - f(x_t)| \leq \epsilon \\ |y_t - f(x_t)| - \epsilon, & \text{otherwise} \end{cases} \tag{3}$$

$$l_2(y_t - f(x_t)) = (y_t - f(x_t))^2. \tag{4}$$

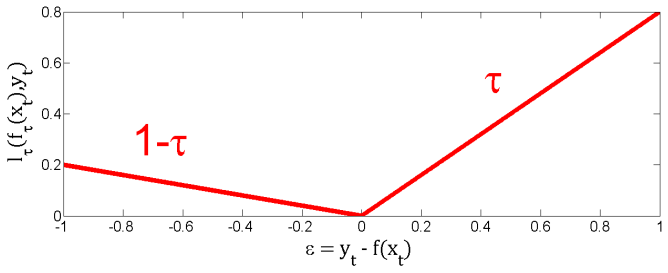


Figure 8: Pinball loss function used for quantile estimation. On the figure τ equals 0.8.

These loss functions do not suit our problem and for that reason we will use another loss function as explained in Section 6.1.1.

6.1.1. Quantile Estimation

The goal we want to achieve is to predict if the chance that a certain SLA will be violated is larger than a predefined value. Suppose, for example, we want to check wether a service has a chance of at least τ to have a response time smaller than the maximal response time $f_{i,max}$. This can be achieved by first obtaining the confidence interval $[0, f_\tau(x_i)]$ such that the chance the response time belongs to this interval equals τ. The service is then selected if $f_\tau(x_i) \leq f_{i,max}$. The true conditional quantile value is denoted as $\mu_\tau(x_i)$ and satisfies

$$\begin{aligned} Pr(y_i < \mu_\tau(x_i)) &\leq \tau \\ Pr(y_i \leq \mu_\tau(x_i)) &\geq \tau. \end{aligned} \tag{5}$$

For simplicity reasons we assume the quantile value is unique (which is not necessary always true).

For a location estimator the mean value minimizes the square error and the median value minimizes the absolute value of the error. Similarly it can be shown the quantile value minimizes the following pinball loss function [12] (Figure 8):

$$l_\tau(y_i - f_\tau(x_i)) = \begin{cases} \tau(y_i - f_\tau(x_i)), & y_i - f_\tau(x_i) \geq 0 \\ (\tau - 1)(y_i - f_\tau(x_i)), & y_i - f_\tau(x_i) < 0. \end{cases} \tag{6}$$

6.2. Performance Indicators

Before designing a quantile estimator we have to know which performance measure(s) we want to minimize out-of-sample. We argue the two performance indicators explained in this Section are important.

6.2.1. Performance Indicator I

To be able compare different quantile estimators we have to quantify how good the estimated quantile value approximates the true quantile value. This quantification is not unique in the sense that two different loss-functions can have an optimum in the true quantile value such that one estimator performs better according to the first loss-function and another estimator performs better according to the second loss-function. That's why, in this Section, we further specify the problem of estimating the quantile value to a problem of minimizing a well chosen cost.

Two types of error can occur in our applications:

- Type I error: a (composite) service is rejected to execute a customers' request in which the actual response time is smaller than the maximal response time ($y_i < f_{i,max}$).
- Type II error: a (composite) service is accepted to execute a customers' request in which the actual response time is larger than the maximal response time ($y_i > f_{i,max}$).

For the first performance indicator we assign costs to both errors: the cost of a type I error equals $1-\tau$ and the cost of a type II error equals τ.

Theorem 1. *If the cost of a type I error equals $1-\tau$ and the cost of a type II error equals τ, then the expected conditional cost for accepting is lower than the expected conditional cost for rejecting if and only if $\mu_\tau(x_i) > f_{i,max}$.*

Proof 1. *The expected conditional cost for accepting a service equals*

$$E(cost|accept, x_i) = Pr(y_i > f_{i,max}|x_i)\tau, \tag{7}$$

where $Pr(y_i > f_{i,max}|x_i)$ equals the probability $y_i > f_{i,max}$, and the expected conditional cost for rejecting a service equals

$$E(cost|reject, x_i) = Pr(y_i < f_{i,max}|x_i)(1-\tau). \tag{8}$$

Accepting is better than rejecting if and only if

$$\begin{aligned} E(cost|accept, x_i) - E(cost|reject, x_i) < 0 &\Leftrightarrow Pr(y_i > f_{i,max}|x_i)\tau - Pr(y_i < f_{i,max}|x_i)(1-\tau) < 0 \\ &\Leftrightarrow Pr(y_i > f_{i,max}|x_i) < 1-\tau \\ &\Leftrightarrow f_{i,max} < \mu_\tau(x_i). \end{aligned} \tag{9}$$

Corollary 1. *If $f_\tau(x_i)$ equals the true conditional quantile value $\mu_\tau(x_i)$ and a service is accepted if and only if $f_\tau(x_i) > f_{i,max}$, then the expected conditional cost is minimized for all possible values of $f_{i,max}$.*

Corollary 1 shows the expected cost can be minimized if one knows the true conditional quantile value. A type I error occurs when $y_i < f_{i,max} < f_\tau(x_i)$ and a type II error occurs when $f_\tau(x_i) < f_{i,max} < y_i$. Given a test set only containing the true response times y_t, we cannot calculate the cost because we have no values for $f_{i,max}$. We need extra assumptions to handle the uncertainty over $f_{i,max}$. The first performance indicator (PI_1) will be defined as the cumulative expected cost, given these assumptions.

Theorem 2. *If we receive one service request per time step, if the probability on a certain f_{max} is time independent and uniformly distributed in an interval $[f^-, f^+]$ with f^-, f^+ finite and if all y_t, $f_\tau(x_t)$ belong to this interval, then the expected cost, given y_t and $f_\tau(x_t)$, becomes proportional to the pinball loss*

$$E(cost|y_t, f_\tau(x_t)) \sim l_\tau(y_t - f_\tau(x_t)). \tag{10}$$

Proof 2. *When $f_\tau(x_t) > y_t$ a type I error (with cost $1-\tau$) can occur and when $y_t < f_\tau(x_t)$ a type II error (with cost τ) can occur. Because f_{max} is uniformly distributed, the probability on these errors becomes proportional to the distance between $f_\tau(x_t)$ and y_t and the expected cost becomes proportional to the pinball loss.*

Corollary 2. *Given the same assumptions as in Theorem 2, the first performance indicator becomes*

$$PI_1(\mathcal{T}, f_\tau) = \sum_{t=1}^{n_{test}} l_\tau(y_{t,test} - f_\tau(x_{t,test})). \tag{11}$$

A disadvantage of using the pinball loss as performance measure is that the loss of one datapoint is not bounded (it can become infinite), despite the fact one datapoint can only cause one error of type I or II. This is caused by the assumption all y_t and $f_\tau(x_t)$ belong to the interval $[f^-, f^+]$ and the interval $[f^-, f^+]$ is thus not a priori determined and can become arbitrary large.

Theorem 3. *If we receive one service request per time step and if the probability on a certain f_{max}, denoted as Pr, is time independent and a priori determined, then the first performance indicator (PI_1) equals the following cumulative expected cost*

$$PI_1(\mathcal{T}, f_\tau, F) = \sum_{t=1}^{n_{test}} l_\tau(y_{t,test}, f_\tau(x_{t,test}); F) = \sum_{t=1}^{n_{test}} l_\tau\left(F(y_{t,test}) - F(f_\tau(x_{t,test}))\right) \tag{12}$$

where F is the cumulative distribution function of Pr

$$F(y) = \int_0^y Pr(f_{t,max}) df_{t,max}. \tag{13}$$

Proof 3. *If $y_{t,test} \geq f_\tau(x_{t,test})$, then an error of type II can occur with probability $F(y_{t,test}) - F(f_\tau(x_{t,test}))$. The expected cost becomes $\tau(F(y_{t,test}) - F(f_\tau(x_{t,test})))$. If $y_{t,test} < f_\tau(x_{t,test})$, then an error of type I can occur with probability $F(f_\tau(x_{t,test})) - F(y_{t,test})$. The expected cost becomes $(1-\tau)(F(f_\tau(x_{t,test})) - F(y_{t,test}))$.*

The influence of one data point on PI_1 is bounded because

$$l_\tau(y_{t,test}, f_\tau(x_{t,test}); F) \leq \max(\tau, 1-\tau). \tag{14}$$

Different probability distributions for f_{max} cause different costs and in practice the performance indicator should equal the cumulative expected cost in which the chosen probability distribution of f_{max} matches the true probability distribution as good as possible. In our experiments we will use the performance indicator as defined in 2 in which the influence of one datapoint is unbounded because we have no data that allows us to make a reasonable estimate of $Pr(f_{max})$.

6.2.2. Performance Indicator II

In practice costs are not always linear in the number of errors. It is possible that causing more violations than agreed invokes huge fines while causing fewer violations than agreed does not cause equally large profits. For that reason it is important the number of violations does not exceed the agreed number of violations to much. Constraining the frequency of violations not to exceed $1-\tau$ is in our opinion to sever because the test set contains only a sample of the total population and it is possible the frequency of violations of this sample exceeds $1-\tau$ despite the frequency of violations of the entire population to be smaller or equal to $1-\tau$. That's why we choose performance indicator II to express the probability the frequency of violations of the test set is larger than or equal to the actual number of violations in the test set, given the the frequency of violations of the entire population equals $1-\tau$ and given certain assumptions.

Theorem 4. *Assuming the expected number of violations equals $1-\tau$ and assuming the data is independent and identically distributed (i.i.d.), the probability of observing at least $n_{v,test}$ violations on a test set where $n_{req,test}$ service requests are accepted becomes*

$$Pr(n_v \geq n_{v,test} | \tau, n_{req,test}) = \sum_{i=n_{v,test}}^{n_{req,test}} (1-\tau)^i \tau^{(n_{req,test}-i)} \binom{n_{req,test}}{i} \tag{15}$$

where $\binom{n}{i}$ is a binomial coefficient and where the number of violations are counted as follows

$$n_{v,test} = \sum_{i=1}^{n_{req,test}} viol(y_{i,test}, f_{i,max}) \text{ where } viol(y_{i,test}, f_{i,max}) \begin{cases} 0, & y_{t,test} \leq f_{i,max} \\ 1, & y_{t,test} > f_{i,max}. \end{cases} \tag{16}$$

Proof 4. *The discrete probability distribution of the number of violations n_v in a sequence of $n_{req,test}$ independent experiments, each of which yields a violation with probability $1-\tau$, equals a binomial distribution. The probability on $n_{v,test}$ or more violations given a binomial distribution is expressed in (15).*

To be able to measure the number of violations, we need test data containing times on which service requests are send together with the corresponding maximal response times. In case no such data is available, we have to make extra assumptions.

Corollary 3. *Under the extra assumptions of one service request per time step and* 100% *acceptance with $f_{i,max}$ equal to $f_\tau(x_i)$ (the latter assumption implies the worst case value of $f_{i,max}$ such that the service is accepted), the probability of observing at least $n_{v,test}$ violations is denoted as PI_2 and equals*

$$PI_2(\mathcal{T}, f_\tau) = \sum_{i=n_{v,test}}^{n_{test}} (1-\tau)^i \tau^{(n_{test}-i)} \binom{n_{test}}{i} \tag{17}$$

where $\binom{n}{i}$ is a binomial coefficient and where the number of violations are counted as follows

$$n_{v,test} = \sum_{t=1}^{n_{test}} viol(y_{t,test}, f_\tau(x_{t,test})) \text{ where } viol(y_{t,test}, f_\tau(x_{t,test})) \begin{cases} 0, & y_{t,test} \leq f_\tau(x_{t,test}) \\ 1, & y_{t,test} > f_\tau(x_{t,test}). \end{cases} \tag{18}$$

The true quantile value of an estimator is defined as

$$\tau_f(f_\tau) = Pr(y_{t,test} < f_\tau(x_{t,test})). \tag{19}$$

Given the violations are counted as in (18), $1-\tau_f(f_\tau)$ is equal to the expected number of observed violations. The smaller PI_2, the more unlikely the observed number of violations are caused by an estimator with true quantile value larger than or equal to τ. A high PI_2, however, does not imply having a conditional quantile estimator. An unconditional, constant function can perform very well on this performance indicator, but will, luckily, perform poorly on PI_1.

6.3. Kernel-Based Quantile Estimation

In this Section we will explain why the kernel-based quantile estimator discussed in [22] suits our problem. The first algorithm we propose is designed to minimize PI_1 and thus the following *expected risk*

$$R[f_\tau] = \int l_\tau \left(y - f_\tau(x)\right) dP(x,y). \tag{20}$$

The probability density function $P(x, y)$ is unknown. We have only access to training data. A naive way of minimizing the *expected risk* is minimizing the *empirical risk*

$$R_{emp}[f_\tau, S] = \frac{1}{n}\sum_{t=1}^{n_{tr}} l_\tau(y_{t,tr} - f_\tau(x_{t,tr})). \tag{21}$$

When the hypothesis space $\mathcal{H}$ to which f_τ belongs is very rich, this can lead to overfitting. For that reason we will minimize the *regularized risk*

$$R_{reg}[f_\tau, S] = R_{emp}[f_\tau, S] + \frac{\lambda}{2}\|w\|_2^2 = \frac{1}{n}\sum_{t=1}^{n_{tr}} l_\tau(y_{t,tr} - f_\tau(x_{t,tr})) + \frac{\lambda}{2}\|w\|_2^2 \tag{22}$$

where λ is a regularization parameter that must be positive. We assume f_τ can be written as

$$f_\tau(x) = w^T\varphi(x) + b \tag{23}$$

where w is the weight vector, φ maps the input space into a feature space (which can be infinite dimensional) and b is the constant offset. Minimizing the *regularized risk* can be written as the following optimization problem

$$\min_{w,b,\xi,\xi^*} \sum_{t=1}^{n_{tr}} \tau\xi_t + (1-\tau)\xi_t^* + \frac{1}{2}\lambda\|w\|_2^2 \text{ subject to } \begin{cases} y_{t,tr} - w^T\varphi(x_{t,tr}) - b \le \xi_t, & t = 1, ..., n_{tr} \\ w^T\varphi(x_{t,tr}) + b - y_{t,tr} \le \xi_t^*, & t = 1, ..., n_{tr} \\ \xi_t, \xi_t^* \ge 0, & t = 1, ..., n_{tr}. \end{cases} \tag{24}$$

Because $\varphi(x)$ can be huge or even infinite dimensional, directly solving this optimization problem can become very expensive. The dual problem, after applying the kernel trick, becomes

$$\min_{\alpha} \frac{1}{2}\lambda\alpha^T\Omega\alpha - \alpha^T y \text{ subject to } \begin{cases} \frac{\tau}{\lambda} \ge \alpha_t \ge \frac{\tau-1}{\lambda}, & t = 1, ..., n_{tr} \\ \sum_{t=1}^{n} \alpha_t = 0 \end{cases} \tag{25}$$

where $k(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$, $\Omega \in \mathbb{R}^{n_{tr} \times n_{tr}}$ is the kernel matrix defined as $\Omega_{i,j} = k(x_{i,tr}, x_{j,tr})$, $\alpha = [\alpha_{1,tr}, ..., \alpha_{n_{tr},tr}]^T$ and $y = [y_{1,tr}, ..., y_{n_{tr},tr}]^T$. This optimization problem is a quadratic programming (QP) problem. The kernel trick ($k(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$) makes explicit calculation of the mapping $\varphi(x)$ into a

possible infinite dimensional feature space no longer necessary. The function f_τ can now be expressed as

$$f_\tau(x) = \sum_{t=1}^{n_{tr}} \alpha_t k(x_{t,tr}, x) + b. \tag{26}$$

The constant offset b can be found using the fact the following counts $f_\tau(x_{t,tr}) = y_{t,tr}$ for α_t belonging to the open interval $]\frac{\tau-1}{\lambda}, \frac{\tau}{\lambda}[$.

The kernel-based quantile estimator explained in this section has some important benefits:

- The estimation is distribution-free in the sense that no assumption on the conditional distribution of the output value given an input vector needs to be made.
- The estimator is robust and thus resistant to outliers. Its breakdown point equals $1 - \tau$ if $\tau \geq 0.5$.
- The estimator ensures, in the batch setting, the quantile curve divides the observations in the desired ratios (τ and $1 - \tau$).

It has important disadvantages as well:

- Different quantile curves can cross each other because each quantile function is estimated independently.
- The estimator is very sensitive to changes in the output values of datapoints near the quantile curves.
- We cannot assure that out-of-sample observations are divided in the desired ratios.

The pseudo-code for the algorithm explained in this section is shown in Algorithm 1.

Algorithm 1 Given the training set $\mathcal{S}$,the targeted quantile value τ, a kernel k and the regularization parameter λ, find α and b which are parameters of the predicting function f_τ.

for $t_1 = 1 \rightarrow n_{tr}$ **do**
  for $t_2 = 1 \rightarrow n_{tr}$ **do**
    $\Omega_{t_1,t_2} \leftarrow k(x_{t_1,tr}, x_{t_2,tr})$
  end for
end for
$\hat{\alpha} \leftarrow \arg\min_\alpha \frac{1}{2}\lambda\alpha^T\Omega\alpha - \alpha^T y_{tr}$

$$\text{subject to } \begin{cases} \frac{\tau}{\lambda} \geq \alpha_t \geq \frac{\tau-1}{\lambda}, & t = 1, ..., n_{tr} \\ \sum_{t=1}^{n} \alpha_t = 0 \end{cases}$$

for $t = 1 \rightarrow n_{tr}$ **do**
  if $\alpha_t \in \ ]\frac{\tau-1}{\lambda}, \frac{\tau}{\lambda}[$ **then**
    $b \leftarrow y_{t,tr} - \sum_{t_1=1}^{n_{tr}} \alpha_{t_1} k(x_{t_1,tr}, x_{t,test})$
  end if
end for

6.4. Online Adaptation of an Unconditional Quantile Estimator

The second algorithm we propose is designed to minimize PI_2. It is a simple online algorithm that is updated as follows

$$f_{\tau,t} = \begin{cases} f_{\tau,start}, & t = 1 \\ f_{\tau,t-1} + \eta\tau, & t > 1, y_{t-1} > f_{\tau,t-1} \\ f_{\tau,t-1} + \eta\left(\tau - 1\right), & t > 1, y_{t-1} < f_{\tau,t-1} \\ f_{\tau,t-1}, & t > 1, y_{t-1} = f_{\tau,t-1}. \end{cases} \tag{27}$$

where η is the learning rate which must be strictly positive. The start value $f_{\tau,start}$ can be set equal to 0, equal to the first data point, or equal to the quantile value of a number of datapoints. Datapoints used to determine $f_{\tau,start}$ should not be used for the on line updating. The number of times y_{t_1} exceeds f_{τ,t_1} for t_1 going from 1 to t is denoted as v_t. We can now express $f_{\tau,t+1}$ as

$$f_{\tau,t+1} = f_{\tau,1} + \eta\left(\tau v_t + (\tau - 1)(t - v_t)\right) = f_{\tau,1} + \eta t\left(\frac{v_t}{t} - (1 - \tau)\right). \tag{28}$$

Conversely, v_t can be expressed in function of $f_{\tau,t+1}$ as follows

$$v_t = \frac{f_{\tau,t+1} - f_{\tau,1}}{\eta} + (1 - \tau)t. \tag{29}$$

Theorem 5. *Assuming there exists an interval $[y_{min}, y_{max}]$ with y_{min}, y_{max} finite such that $f_{\tau,start}$ and all y_t belong to this interval, the frequency of violations converges to $1 - \tau$ for t going to ∞*

$$\lim_{t\to\infty} \frac{v_t}{t} = 1 - \tau. \tag{30}$$

Proof 5. *Because all y_t belong to the interval $[y_{min}, y_{max}]$, all $f_{\tau,t}$ belong to the interval $[y_{min}+\eta\left(\tau - 1\right), y_{max}+\eta\tau]$ and*

$$\begin{aligned} &|f_{\tau,t+1} - f_{\tau,1}| \leq C \\ \Rightarrow &\eta t|\frac{v_t}{t} - (1 - \tau)| \leq C \\ \Rightarrow &|\frac{v_t}{t} - (1 - \tau)| \leq \frac{C}{\eta t} \end{aligned} \tag{31}$$

where $C = y_{max} - y_{min} + \eta\max(\tau, 1 - \tau)$. One can notice the distance between $\frac{v_t}{t}$ and $(1 - \tau)$ converges to zero for t going to ∞.

Theorem 6. *Assuming all y_t belong to the interval $[y_{min}, y_{max}]$ with y_{min}, y_{max} finite, the performance indicator PI_2 of the test set converges to 0.5 for $t \to \infty$.*

Proof 6. *Suppose X_t is a binomial distributed random variable where $1-\tau$ is the number of violations in a sequence containing t datapoints. One can notice that PI_2 expresses the probability the number of violations is at least v_t for this distribution. The mean and the standard deviation of the distribution equal*

$$\mu_{X,t} = t(1-\tau) \tag{32}$$

$$\sigma_{X,t} = \sqrt{t\tau(1-\tau)}. \tag{33}$$

From (31) we can now conclude

$$\frac{|v_t - \mu_{X,t}|}{\sigma_{X,t}} \leq \frac{C}{\eta\sqrt{t\tau(1-\tau)}}, \tag{34}$$

which implies

$$\lim_{t\to\infty} \frac{v_t - \mu_{X,t}}{\sigma_{X,t}} = 0. \tag{35}$$

According to the central limit theorem*, the cumulative distribution function (cdf) of X_t converges pointwise to the cdf of a normal distribution with mean $\mu_{X,t}$ and standard deviation $\sigma_{X,t}$ for n going to ∞. Because, as n approaches ∞, v_t is distanced 0 standard deviations from $\mu_{X,t}$, the cdf of X_t in v_t equals 0.5 and PI_2 becomes 0.5.*

The pseudo-code for the algorithm explained in this section is shown in Algorithm 2.

Algorithm 2 Given x_{test}, y_{test}, η and $f_{\tau,start}$, calculate $f_{\tau,t}(x_{t,test})$ and update $f_{\tau,t}$ for $t = 1, ..., n_{test}$.

```
f_{τ,1} ← f_{τ,start}
for t = 1 → n_test do
  if y_{t,test} > f_{τ,t} then
    f_{τ,t+1} ← f_{τ,t} + τη
  else if y_{t,test} < f_{τ,t} then
    f_{τ,t+1} ← f_{τ,t} + (τ − 1)η
  else
    f_{τ,t+1} ← f_{τ,t}
  end if
end for
```

6.5. Kernel-Based Quantile Estimation with Online Adaptation of the Constant Offset

In this section we will present the final algorithm that combines the algorithms explained in the previous two Sections. Additionally we define the input vectors and explain how the data is preprocessed. The final algorithm consists of three stages.

- Preprocessing phase: in this phase the inputs and corresponding outputs are generated from the response times RT_t. We will predict the logarithm of response times instead of the response times themselves because this avoids our algorithm to make predictions below zero. The input vectors can be past response times (autoregressive), the quantile values of sets of past response times or the time at which the service starts. The best choice for the input vectors depends on the specific dataset one wants to make predictions upon.

- Training phase: Secondly we optimize α and b on the training set using the algorithm explained in Section 6.3 to predict the logarithm of the response time.

- Online updating phase: finally we try to predict $\log(y_t) - f_\tau(x_t)$ with a function denoted as $d_{\tau,t}$ using the algorithm as explained in Section 6.4. The optimal $d_{\tau,start}$ can be determined using a validation set or can be set to 0. We add d_t to $f_\tau(x_t)$ to make the final prediction. The latter is equivalent to updating the constant offset b_t such that b_t equals $b + d_t$.

This final algorithm performs well on both PI_1 and PI_2.

7. Experimental Evaluation

We limit the experimental evaluation to predicting the response time. In a first scenario we predict the response time of an individual service, in a second scenario we predict the response time of an entire workflow.

7.1. Prediction For Individual Services

The kernel-based quantile estmiator (KQ) explained in Section 6.3 and the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) explained in Section 6.5 will be evaluated using a real-

Algorithm 3 Given the true response times RT_t for $t = 1, ..., n + L$ generate the input vectors x_t and the output values y_t for $t = 1, ..., n$. RT_t is the true response time at time t, x_t is the input vector at time $t + L$ and y_t is the output vector at time $t + L$. The function $quantile\,(\tau, \mathcal{X})$ expresses the τ-quantile value of the set $\mathcal{X}$.

for $t = 1 \rightarrow n$ **do**
  $q_{1,t} \leftarrow quantile\left(\tau, \{RT_t\}_{i=t}^{t+\frac{L}{4}-1}\right)$
  $q_{2,t} \leftarrow quantile\left(\tau, \{RT_t\}_{i=t+\frac{L}{4}}^{t+\frac{L}{2}-1}\right)$
  $q_{3,t} \leftarrow quantile\left(\tau, \{RT_t\}_{i=t+\frac{L}{2}}^{t+\frac{3L}{4}-1}\right)$
  $q_{4,t} \leftarrow quantile\left(\tau, \{RT_t\}_{i=t+\frac{3L}{4}}^{t+L-1}\right)$
  $x_t \leftarrow [\log(q_{1,t}), \log(q_{2,t}), \log(q_{3,t}), \log(q_{4,t})]^T$
  $y_t \leftarrow \log(RT_{t+L})$
end for

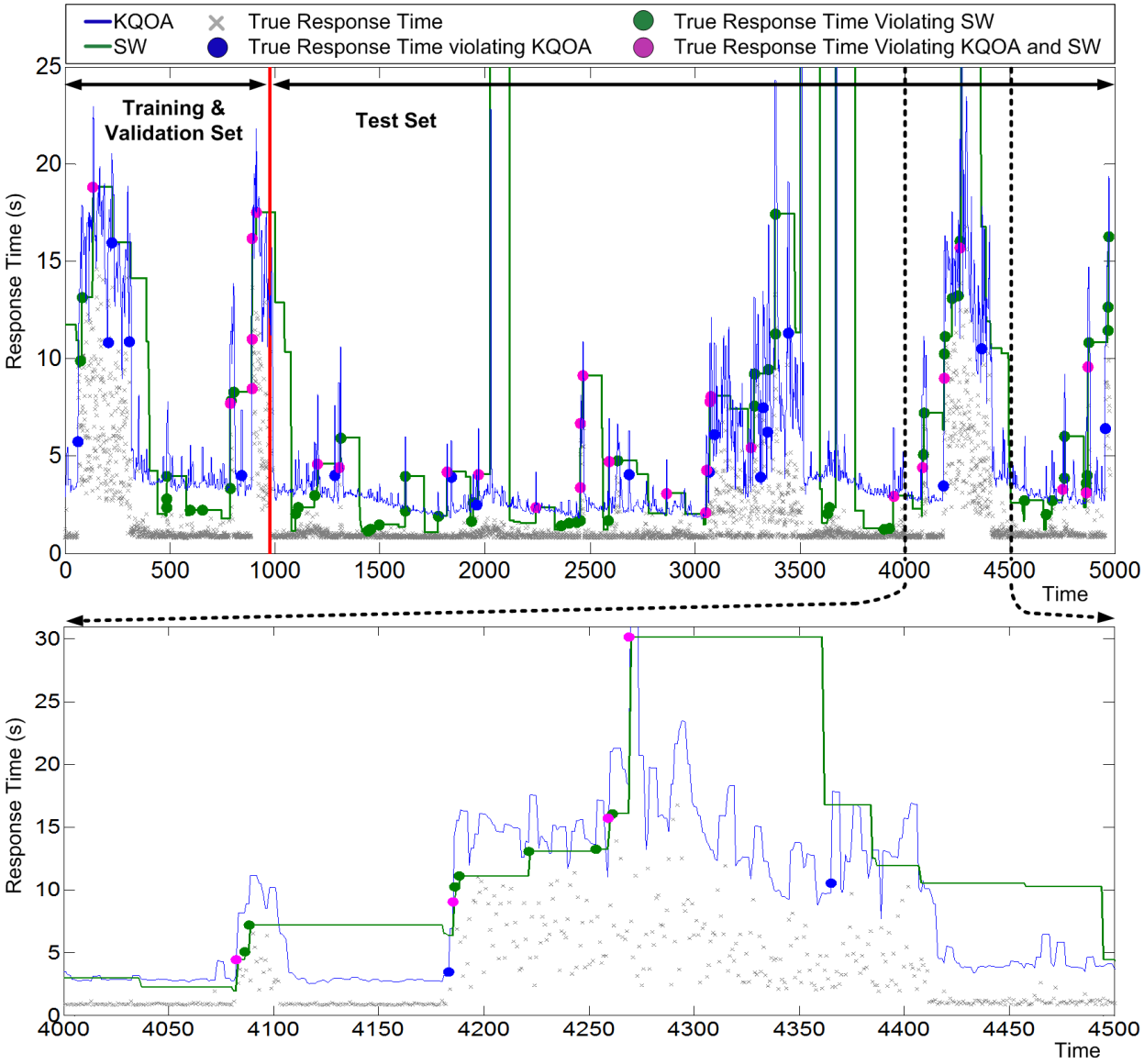


Figure 9: Quantile estimation for the Movie Dataset using the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) and the sliding window estimator (SW). The upper figure shows the results for data point 0 to 5000. The lower figure zooms in on data point 4000 to 4500.

life data set discussed in Section 3.5: the Movie Dataset . We also compare the performance of our algorithms to a sliding window approach, which is used in related work [17].

The kernel we use is a RBF-kernel. The input vectors are quantile values of sets of past response time as shown in Algorithm 3. KQ has three hyperparameters: the regularization parameter λ, the kernel parameter σ and the lag L. KQOA has an extra parameter: the online learning rate η. The training set consists of the first 700 datapoints and will be used to find the optimal α and b, given the hyperparamaters. The validation set consists of the next 300 datapoints and will be used to optimize the hyperparameters λ, σ and L. The

(a) Movie Dataset

	PI_1	$PI_2(\%)$	$f_{v,test}$
KQOA	0.0635	58.5	0.98
KQ	0.0637	99.6	0.63
SWQ	0.0974	0.0	1.85

(b) Adjusted Movie Dataset

	$PI_2(\%)$	$f_{v,test}$
KQOA	6.9	1.25
KQ	0.0	1.75
SWQ	0.0	2.95

Table 6: Comparison of the performance of the Kernel-based Quantile Estimator with Online Adaptation of Constant Offset (KQOA) and the Sliding Window Quantile Estimator (SWQ) on the Movie Dataset (a) and the Movie Dataset with an increase of 15s in response time of 50 randomly selected test set datapoints (b). PI_1 equals the cumulative pinball loss, $PI_2(\%)$ is the probability an estimator with true quantile value 99% causes equal or more violations and $f_{v,test}$ is the number of violations in the test set per 100 datapoints.

learning rate η of the second algorithm allows adaptation to unexpected events and should not be chosen to small. Unexpected behavior that might happen in the test set might not happen in the validation set. We have chosen η as big as possible such that the performance according to performance indicator I on the validation set does not increase with more than 2%.

When the optimal hyperparameters are found, the algorithm is retrained on both the training and the validation set. The performance will then be evaluated using the test set which contains the remaining 4000 datapoints. We compare our algorithms to a sliding window quantile estimator with windowsize w. The latter estimator uses the τ-quantile value of the lasts w datapoints as τ-quantile estimation for the next datapoint. The parameter w is optimized using the first 1000 datapoints as training set and evaluated using the next 4000 datapoints as test set. The KQOA and SW quantile estimations for the Movie Dataset are shown in Figure 9. The lower figure zooms in on the area between data point 4000 and 4500. The performances of the estimators for a quantile value of 99% are summarized in Table 6(a) . According to PI_1 the expected cost using the kernel-based algorithms is approximately 30% lower compared to the sliding window algorithm. According to PI_2 the probability an estimator with true quantile value 99% causes equal or more violations than the estimator is quite big for the kernel-based algorithms and almost non-existing for the sliding window estimator.

We can conclude our algorithms performs significantly better than the sliding window estimator on both PI_1 and PI_2. Without online adaptation, however, we cannot guarantee that PI_2 will not converge to zero. With online adaptation, given certain assumptions, we can. To show the effect of the online adaptation, we increased the response time of 50 randomly selected datapoints of the test set by 15 seconds. This causes the KQ algorithm to perform bad since these spikes only appear in the test set and are thereby not learned during the training period. In Table 6(b), we can see that for this experiment only KQOA has a value for PI_2 that differs significantly from zero. The optimal hyperparameters for our algorithms are: $L = 24$, $\sigma = 500$,

$\lambda = 0.005$ and $\eta = 0.1$. The optimal window size for the sliding window quantile estimator equals 92.

7.2. Prediction For Composite Services

For predicting the response time of composite services, we need to calculate the simulated response times according to the technique explained in Section 5. We use the e-health case study described in Section 3.1, where each elementary service has a response time according to simulated data for a long-running process. Rosario [17] validated the use of certain families of distributions and performed their best fit on measured data real-life web services. They observed that the Gamma and the Log-Logistic distributions were a reasonably good fit for the response times.

To simulate data for the elementary services, we will assume the number of working hours needed to complete a task is modeled as log-logistic distributed random variables. The log-logistic distribution is similar in shape to the log-normal distribution but has heavier tails. The probability density function depends on α and β:

$$f(x;\alpha,\beta) = \frac{(\frac{\beta}{\alpha})(\frac{x}{\alpha})^{\beta-1}}{[1+(\frac{x}{\alpha})^{\beta}]^2}. \tag{36}$$

Some of the services in our case study require human interaction and thus we assume the completion time of a task is subject to working hours of the service. Depending on the service, a working day starts and ends at a specific hour and are open or closed on Saturdays and Sundays. An overview of our assumptions is shown in Table 7. Outside the range of working hours the progress of the task is freezed. For example, if a working day starts at 8h and ends at 16h, then a task taking 9 working hours to complete that starts at 10h on Monday will end at 11h on Tuesday. The response time of 25 hours consist of 6 working hours within the same day, 16 non-working hours and 3 working hours the next day. We assume monitoring is done every hour. The generated response times for all elementary services of the e-health workflow are shown in Figure 10. The graph at the bottom represents the resulting composite response time, calculated using the PNET-system described in Section 5.

We will compare the Kernel-Based Quantile estimator (KQ) explained in Section 6.3 and the Kernel-Based Quantile estimator with Online Adaptation of the constant offset (KQOA) explained in Section 6.5 to the probabilistic approach using Boostrap-Based Simulations explained in the paper of Rosario [17]. In this work, they use bootstrap based Monte Carlo (BBMC) simulations to estimate a composite QoS value by repeatedly drawing QoS values from the probability distributions of the constituting elementary services and composing them using soft contract rules. The training set consists of the first 4 weeks, the validation set

Services	Parameters				
	Start	**End**	**Weekend**	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$
Post 1	8h	15h	Closed	log(10)	1/8
Radiology	9h	18h	Closed	log(14)	1/10
Screening 1	8h	20h	Saturday	log(12)	1/9
Screening 2	8h	22h	Closed	log(12)	1/7
Screening 3	8h	20h	Saturday	log(15)	1/7
Report	0h	24h	Open	log(2)	1/10
Billing	0h	24h	Open	log(5)	1/4
Post 2	8h	15h	Closed	log(10)	1/8

Table 7: Parameter assumptions for elementary services of e-health case study

(used to optimize hyperparameters) of the next 4 weeks and the test set consists of the last 8 weeks. After optimizing the hyperparameters the algorithm is retrained on both the validation and training set. BBMC uses past response time as input which has the problem that recent executions of individual services may not have ended yet. Their response time is thus unknown. We solve this by sliding the window backwards in time until all response times within and before the window are known. As shown in Algorithm 4, we take the time (in hours) within a week as input vector for KQ and KQOA, because we a priori know there is weekly seasonality. The KQOA and BBMC quantile estimations for the e-health application are shown in Figure 11. Table 8 shows KQ and KQOA perform better than BBMC and shows the number of violations for BBMC is very low. The estimated quantile value of BBMC is thus an overestimate of the true quantile value which can be explained as follows:

For some services the progress freezes during the weekends. The response time of these services on friday evening will be very high. Suppose we have two services whose response time are shown in Table 9 that need to be executed sequentially. The number of working hours to complete service 1 is exactly 2 and the number

	PI_1	$\text{PI}_2(\%)$	$f_{v,test}$
KQOA	0.795	69.1	0.89
KQ	0.857	3.3	1.56
BBMC	1.212	100.0	0.0

Table 8: Comparison of the performance of the Kernel-based Quantile Estimator (KQ), Kernel-based Quantile estimator with Online Adaptation of Constant Offset (KQOA) and the probabilistic approach using Boostrap-Based Simulations (BBMC) on the E-Health Dataset. PI_1 equals the cumulative pinball loss, $\text{PI}_2(\%)$ is the probability an estimator with true quantile value 99% causes equal or more violations and $f_{v,test}$ is the number of violations in the test set per 100 datapoints. The chosen hyperparameters for KQOA and/or KQ are: $\sigma = 50$, $\lambda = 0.01$ and $\eta = 0.02$. The chosen window-size for BBMC equals 300.

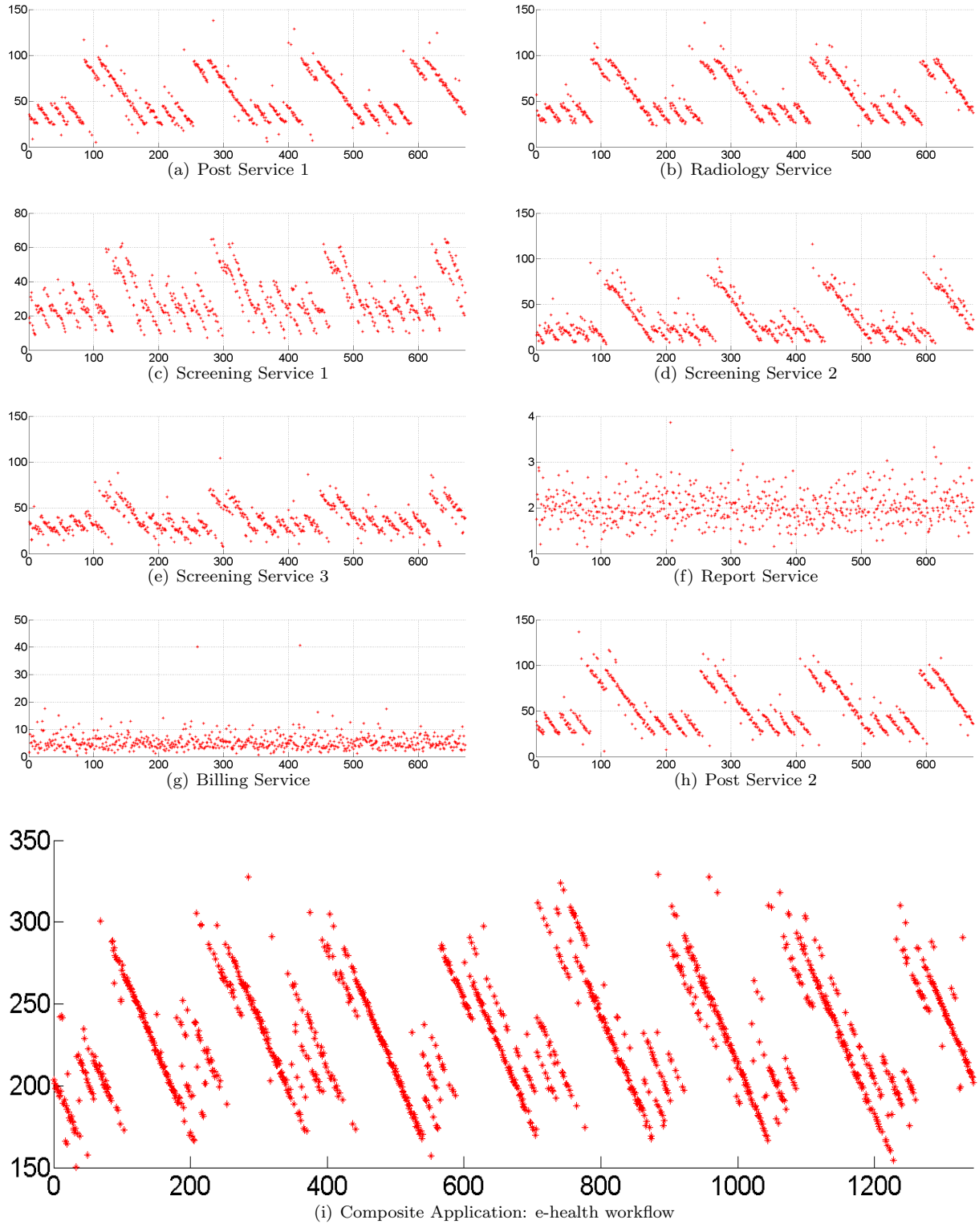


Figure 10: Overview of response times of elementary service (a-h) and resulting simulated response time of composite service (i) for e-health case-study (horizontal axis: time (in hours) - vertical axis: response time (in hours))

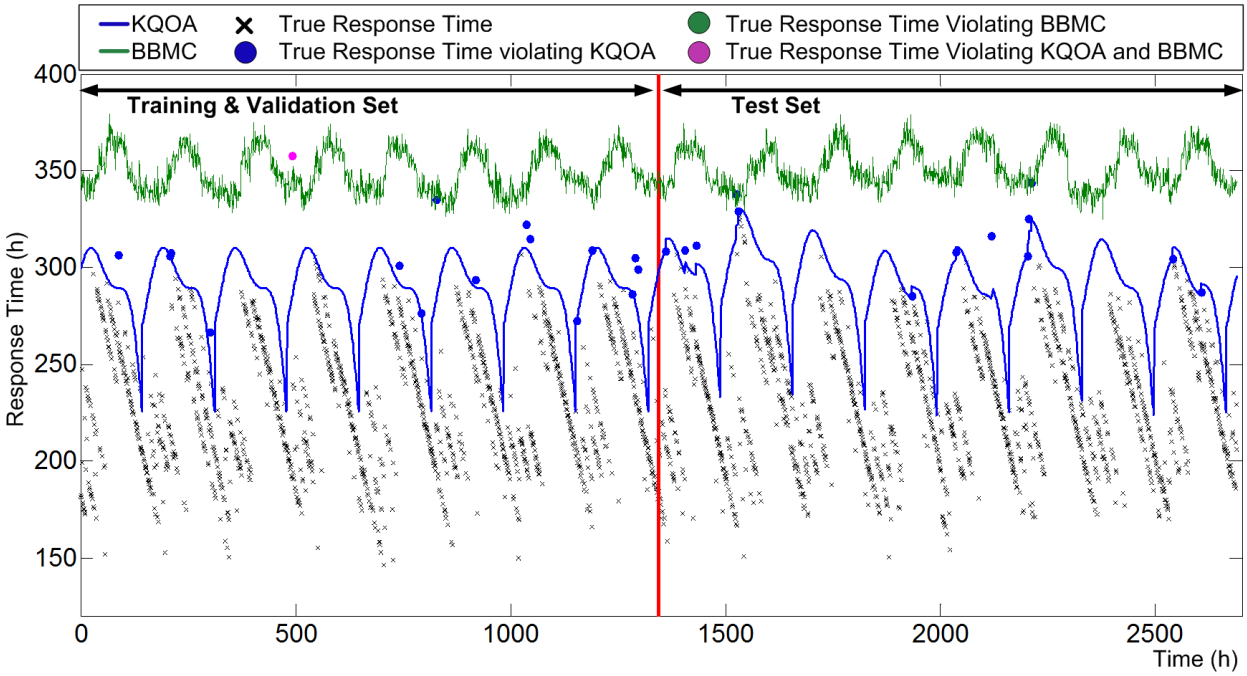


Figure 11: Quantile estimation for the e-health application using the kernel-based quantile estimator with online adaptation of the constant offset (KQOA) and the bootstrap based Monte Carlo estimator (BBMC).

of working hours needed to complete service 2 is exactly 3. The working day starts at 8h and ends at 16h. The time to complete this workflow on friday is 5 hours until 11h and 69 hours (5 hours + 16 hours + 48 hours) after 11h. If, during the Monte Carlo Simulation the response time at 16h of service 1 (which equals 66 hours) and the response at 15h of service 2 (which equals 67 hours) are combined, then this causes a total response time of 133 hours. The response time will never be that high in practice. The worst case scenario using Monte Carlo thus becomes worse than the 'actual' worst case scenario which causes the estimated 99%-quantile value to be too high. For a similar reason it can be shown the estimated 1%-quantile values on the E-Health dataset are on average lower than the true 1%-quantile values. The approach used in KQ and KQOA of first composing the response times as explained in Section 5 and then estimating the quantile values avoids this problem. Another reason for the poor performance of BBMC is that it cannot learn the seasonality patterns.

Time(hours)	8	9	10	11	12	13	14	15	16
RT_1	2	2	2	2	2	2	2	66	66
RT_2	3	3	3	3	3	3	67	67	67

Table 9: Response times of two services on Friday. For both services the progress freezes during the weekend. The response time of service 1 is denoted as RT_1 and the response time of service 2 is denoted as RT_2.

Algorithm 4 Given the true response times RT_t for $t = 1, ..., n$ generate the input vectors x_t and the output values y_t for $t = 1, ..., n$.

for $t = 1 \rightarrow n$ **do**
  $x_t \leftarrow t \mod 24 \cdot 7$
  $y_t \leftarrow RT_t$
end for

8. Conclusion & Future Work

In this paper we developed a two-step approach to predict QoS values of composite services. The first step, which is based upon Petri nets, derives simulated QoS-values of a workflow composition from those of its constituent elementary services. In a second step we focus on the response time and try to predict future response times of composite services based upon the simulated response times using a kernel-based quantile estimator with online adaptation of the constant offset. The latter algorithm has a batch and an online learning phase. During the batch learning phase it tries to minimize, in a regularized manner and given certain assumptions, the occurrence of two types of errors: a type I error occurs when a service is rejected in which the real response time is smaller than the agreed response time and a type II error occurs when a service is accepted in which the real response time exceeds the agreed response time. The online learning phase is necessary to assure that the probability an estimator with true quantile value equal to the agreed quantile value causes equal or more violations, converges to 50% for the number of test datapoints going to infinity. The latter implies the number of times the true response time exceeds the estimated response time converges to the agreed quantile value.

We evaluated our prediction algorithm on one elementary service, an automated service using real-life web service data, and on one composite service, a service with human interaction using simulated data. On the first dataset our algorithm caused a loss 30% lower than a sliding window estimator and on the second dataset our algorithm significantly outperformed the probabilistic approach using Boostrap-Based Simulations (BBMC) explained in the paper of Rosario et al. [17].

Our future work includes a practical implementation of QoS-aware service composition using the algorithms presented in this paper. Currently we are evaluating our approach in the context of Cloud Computing. Software as a service (SaaS), sometimes referred to as "Software on Demand", is an emerging mechanism of releasing software applications to customers. From a technical perspective, an import issue in "software on demand" is the customization for the SaaS application to serve multiple tenants, each having having their own service level requirements. Our goal is to have a practically efficient solution to address customization of business processes with a focus on run-time assurance of service quality and service level agreements. For the implementation we use the framework discussed in [5]. The framework extends the WS-BPEL language

with policy-based reconfiguration capabilities. The framework is based on the Model-View-Controller (MVC) pattern and is implemented in Ruby On Rails (RoR). To address the challenge of functional based service selection, as mentioned in Section 2, we have a master thesis running on extending the framework with semantic based service selection support. To achieve accurate estimates of the response times of the individual services and extend them to an estimate for the composite service, we will use the techniques described in this work. The logic to efficiently select appropriate services among available services will be implemented using the reconfiguration support of the framework. A challenge here is to implement an efficient selection algorithm that takes into account the stochastic behavior of the QoS estimates of candidate services by integrating the QoS estimation techniques presented in this work.

References

[1] E. Al-Masri, Q. H. Mahmoud, Investigating web services on the world wide web, In: Int. Conf. on the World Wide Web, WWW 2008, New York, USA, 2008, pp. 795-804.

[2] D. Ardagna, B. Pernici, Global and local QoS constraints guarantee in web service selection, In: IEEE Int. Conf. on Web Services, ICWS, 2005, pp. 805806. IEEE Comput. Soc.

[3] G. Canfora, M. D. Penta, R. Esposito, M. L. Villani, An approach for QoS-aware service composition based on genetic algorithms, In: Genetic and Evolutionary Computing Conf., GECCO 2005, Washington DC, USA, 2005, pp. 1069-1075.

[4] J. Cardoso, Quality of Service and Semantic Composition of Workflows, PhD thesis, Univ. of Georgia, 2002.

[5] K. Geebelen, E. Kulikowski, E. Truyen, W. Joosen, A MVC Framework for Policy-Based Adaptation of Workflow Processes: A Case Study on Confidentiality, In: IEEE Int. Conf. on Web Services, ICWS 2010, 2010, pp. 401-408.

[6] M. Gillmann, G. Weikum, W. Wonner, Workflow management with service quality guarantees, In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 228239.

[7] J. Harney, P. Doshi, Speeding up adaptation of web service compositions using expiration times, Int. Conf. on World Wide Web, WWW 2007, New York, USA, ACM, 2007, pp. 1023-1032.

[8] S.-Y. Hwang, H. Wang, J. Tang, J. Srivastava, A probabilistic approach to modeling and estimating the QoS of web-services-based workflows, Information Sciences 177 (2007) 54845503.

[9] M.C. Jaeger, G. Rojec-Goldmann, G. Mhl, QoS aggregation for Web service composition using workflow patterns, In: Proceedings of IEEE 8th Int. Conf. on Enterprise Distributed Object Computing, EDOC04, 2004, pp. 149159.

[10] B. Kiepuszewski, A.H.M. ter Hofstede, C. Bussler, On Structured Workflow Modelling, In: Proc. of the 12th Int. Conf. on Advanced Information Systems Engineering, CAISE00, 2000, pp. 431-445.

[11] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Trans. on Signal Processing 52 (8) (2002) 2165-2176.

[12] R. Koenker, G. Bassett, Regression quantiles, Econometrica 46 (1) (1978) 33-50.

[13] D. Mukherjee, P. Jalote, M. Nanda, Determining QoS of WS-BPEL Compositions, In: Int. Conf. on Service-Oriented Computing, ICSOC 2008, Springer, Heidelberg, 2008, pp. 378-393.

[14] S. Narayanan, S. McIlraith, Simulation, verification and automated composition of web services, In: Proc. of Int. World Wide Web Conf., WWW 2002, Honolulu, Hawaii, 2002, pp. 7788.

[15] M. Papazoglou, K. Pohl, Report on Longer Term Research Challenges in Software and Services, Results from two workshops held at the European Commission premises at 8th of November 2007 and 28th and 29th of January 2008, with contributions from M. Boniface, S. Ceri, M. Hermenegildo, P. Inverardi, F. Leymann, N. Maiden, A. Metzger, T. Priol, European Commission, `http://www.cordis.lu`, 2008.

[16] P. Puschner, A. Burns, A Review of Worst-Case Execution-Time Analysis (Editorial), Real-Time Systems 18 (2/3) (2000).

[17] S. Rosario, A. Benveniste, S. Haar, C. Jard, Probabilistic QoS and Soft Contracts for Trans. based Web Services, IEEE Trans. on Services Computing 1 (4) (2008) 187200.

[18] K. Salimifard M. Wright, Petri Net-based modeling of workflow systems: an overview. Eur. J. Oper. Res. 134 (3) (2001) 664676.

[19] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, H. Mei, Personalized QoS Prediction for Web Services via Collaborative Filtering, In: IEEE Int. Conf. on Web Services, ICWS 2007, 2007, pp. 439-446.

[20] A. J. Smola, B. Schlkopf, A tutorial on support vector regression, Statistics and Computing 14 (2004) 199-222.

[21] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Pub. Co., Singapore, 2002.

[22] I. Takeuchi, Q.V. Le, T.D. Sears, A.J. Smola, Nonparametric Quantile Estimation, J. of Machine Learning Research 7 (2006) 1231-1264.

[23] V. Vapnik, Statistical learning theory, John Wiley & Sons, New York, 1998.

[24] H.-C. Wang, C.-S. Lee, T.-H. Ho, Combining subjective and objective QoS factors for personalized Web service selection, Expert Systems with Applications 32 (2) (2007) 571584.

[25] W. Wiesemann, R. Hochreiter, D. Kuhn, A Stochastic Programming Approach for QoS-Aware Service Composition, In: Int. Symposium on Cluster Computing and the Grid, CCGrid 2008, Lyon, France, May 2008, pp. 226-233.

[26] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-aware middleware for web services composition, IEEE Trans. on Software Engineering 30 (5) (2004) 311-327.



Dries Geebelen received a degree in civil engineering from the K.U.Leuven in 2009. He is PhD student in the department of Electrical Engineering of the K.U.Leuven. His research interests include kernel-based methods, network modeling and non-parametric quantile estimation.



Kristof Geebelen is a PhD student within the DistriNet research group of the K.U.Leuven. His main research interests are Service Oriented Architectures, dynamic workflow adaptation, security frameworks and Aspect Oriented Software Development. Since Kristof started at the Department of Computer Science in 2005, he worked on several projects including WIT-CASE, T-CASE, COCOMEDIA and CUPID.



Eddy Truyen is a senior researcher at DistriNet where he is responsible for leading the research of the DistriNet task force on Language technology and Middleware. His main research interests include aspect-oriented middleware, middleware for service-based architectures, dynamic reconfiguration, and support for building reusable aspect frameworks.



Sam Michiels obtained his PhD in 2003 at the DistriNet research group at K.U.Leuven where he was appointed as Industrial Research Manager in 2008. Sam is responsible for research and technology transfer in the domain of distributed software systems. His research interests include middleware, component frameworks and software management of large-scale distributed systems such as wireless sensor networks and the cloud.



Johan A. K. Suykens was born in Willebroek Belgium, May 18 1966. He received the degree in Electro-Mechanical Engineering and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor (Hoogleraar) with K.U.Leuven. He is author of the books "Artificial Neural Networks for Modelling and Control of Non-linear Systems" (Kluwer Academic Publishers) and "Least Squares Support Vector Machines" (World Scientific), co-author of the book "Cellular Neural Networks, Multi-Scroll Chaos and Synchronization" (World Scientific) and editor of the books "Nonlinear Modeling: Advanced Black-Box Techniques" (Kluwer Academic Publishers) and "Advances in Learning Theory: Methods, Models and Applications" (IOS Press). In 1998 he organized an International Workshop on Nonlinear Modelling with Time-series Prediction Competition. He is a Senior IEEE member and has served as associate editor for the IEEE Transactions on Circuits and Systems (1997-1999 and 2004-2007) and for the IEEE Transactions on Neural Networks (1998-2009). He received an IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as a program co-chair for the International Joint Conference on Neural Networks 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007 and a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning.



Joos Vandewalle was born in Kortrijk, Belgium, in August 1948. He obtained the electrical engineering degree and a doctorate in applied sciences, both from the Katholieke Universiteit Leuven, Belgium in 1971 and 1976 respectively. From 1976 to 1978 he was Research Associate and from July 1978 to July 1979, he was Visiting Assistant Professor both at the University of California, Berkeley. Since July 1979 he is back at the Department of Electrical Engineering (ESAT) of the Katholieke Universiteit Leuven, Belgium where he is Full Professor since 1986 and the head of the SCD division at ESAT, which has more than 130 researchers. From August 1996 till August 1999 and from August 2003 till February 2005 he was Chairman of the Department of Electrical Engineering. From August 1999 till July 2002 he was the vice-dean of the Faculty of Engineering at the Katholieke Universiteit Leuven. In the second semester of 2002-2003 he was on sabbatical leave at the I3S laboratory of CNRS Sophia Antipolis, France. From February 2005 on he is a member of the Governing Board of Science & Technology at the K.U.Leuven.

He teaches courses in linear algebra, linear and nonlinear system and circuit theory, system identification and neural networks. His research interests are mainly in mathematical system theory and its applications in circuit theory, control, signal processing, cryptography and neural networks. His recent research interests are in nonlinear methods (support vector machines, multilinear algebra) for data processing. He has authored or coauthored more than 300 international journal papers in these areas. He is the co-author of 4 books and co-editor of 5 books. He is a member of the editorial board of the International Journal of Circuit Theory and its Applications, Neurocomputing, Neural Networks and the Journal of Circuits Systems and Computers. From 1989 till 1991, he was associate editor of the IEEE Transactions on Circuits and Systems. He was Deputy Editor-in-chief of the IEEE Transactions on Circuits and Systems part I Fundamental theory and applications from January 2002 till December 2003. Since 2001 he is a member of the Advisory Board of the International Journal on Information Security (IJIS). He was program chairman of ISCAS 2000 in Geneva, and for IJCNN 2004 in Budapest. He was Chairman of the NOLTA conference in Bruges in 2005. He was elected Fellow of IEEE in 1992 for contributions to nonlinear circuits and systems and in 2006 as Vice-President Technical Activities of the IEEE CAS Society. In 1991-1992 he held the Francqui chair on Artificial Neural Networks at the University of Lige and in 2001-2002 he held this chair on Advanced Data Processing techniques at the Free University of Brussels. He is also Fellow of the IEE (UK). He received several best paper awards and research awards. He is a member of the Academia Europaea and of the Belgian Academy of Sciences and its CAWET committee on engineering. He is a member of the Technology Committee of the Vlaamse Raad voor WetenschapsBeleid (VRWB).



Wouter Joosen is full professor at the Department of Computer Science of the Katholieke Universiteit Leuven in Belgium, where he teaches courses on software architecture and component-based software engineering, distributed systems and the engineering of secure service platforms. His research interests are in aspect-oriented software development, focusing on software architecture and middleware, and in security aspects of software, including security in component frameworks and security architectures.