Predicting Service Request Rates for Adaptive Resource Allocation in SOA

Alexander Serebrenik and Natalia Sidorova

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{a.serebrenik,n.sidorova}@tue.nl

Abstract. Service orientation is rapidly becoming the common practice in the IT world. A price one often has to pay for the advantages of service oriented architectures (SOA) is performance deterioration. SOA performance heavily depends on the allocation of computational resources to services. The needs of services in computational resources are however changing, depending e.g. on the environmental factors and changes in business processes (and hence service orchestrations). To ensure good performance results, the resource allocation should respond to the changes in the SOA environment. In this paper we focus on the detection of the changes in the environment and the prediction of the expected service requests rates. For this purpose we first discover a stochastic model of the service request rates. Then we monitor the system to detect changes in the environment behaviour and signal the necessity to reconsider the resource allocation, providing a prediction of the service request rates for the coming period. Moreover, we monitor whether the model is still a fair reflection of the behaviour, and when necessary, we adapt the model appropriately.

Key words: service, performance, adaptivity, statistical analysis

1 Introduction

Service orientation [15, 28] is almost unanimously proclaimed to become the common practice in the IT world within a couple of years. One of the most essential benefits of service orientation for business is delivering enterprise agility. A promise of service reconfiguration flexibility, with changes done in days by business people, not in weeks by technical specialists attracts many enterprises on the SOA (service-oriented architectures) side. Among other advantages are reusability raised to the new level, namely reusability of services as business units of logic, lowering costs and risks of software evolution.

Today's SOA practice at many enterprises is still very far from the "ideal" one, as introducing SOA often results in performance deterioration, i.e., implementation of SOA often compromises quality of service that should be ensured

by the service-level agreement. Indeed, compared to "bare" functionality implementation the use of services implies computation overheads pertaining to inter-service communication via XML messages, service composition, orchestration and invocation. Moreover, although one of the most important SOA principles is that software modules are independent of implementation and infrastructure details, SOA performance heavily depends on the allocation of computational resources to services [27]. To optimise the performance by balancing the resource load and minimising data transfer time, resource allocation should take into account service requests rates and the volumes of data exchanged by services.

Optimization of the resource allocation is difficult already due to the distributed and autonomous nature of services. Among other factors complicating the optimization are factors related to the changing nature of the SOA environment:

- business processes that should be supported by SOA are subject to frequent changes due to the evolving nature of businesses, which is reflected in the changes of service orchestrations or/and implementations;
- even when the business processes supported by SOA remain relatively stable, the amount of work assigned to different services can vary depending on external factors such as season, day time, market conditions, etc.

Due to this changing nature, the resource allocation for SOA providing the best performance does not exist in principle.

To ensure good performance results, the resource allocation should respond to the changes in the SOA environment. Therefore, we want to automate the detection of the changes and the prediction of the expected service requests rates. We stress that we consider only static SOA, i.e., in this paper we do not consider the resource availability problem. This is, for instance, the case for an intra-enterprise SOA or many Software as a Service (SaaS) applications.

To provide an efficient resource allocation, we need not only to see the current snapshot of the system with the information about the resource load, the rate of service requests, etc., but also to be able to predict the developments that can be expected in the near future. The importance of adaptivity has been recognized both by the scientific community [28] and by industry [25].

To provide the input data for the adaptive resource allocation mechanisms, we build the Adaptive Predictive Model (APM) capturing the expected system behaviour. The APM provides for the ability to signal the necessity to re-allocate resources before a critical workload of resources is achieved.

Every state of the APM reflects the characteristics of the system behaviour in some work mode. The changes of modes are triggered by changes in the environmental factors influencing the system, e.g. the start of a holiday period, weather conditions, changes in the stock exchange indices, etc. Some of these factors are known *a priori* as potentially influencing the system and they can be monitored using (external) information sources. Other factors can only be seen via the increase/decrease of the amount of requests of some particular service, which can be monitored on the system. When a change of the mode is observed or expected, the APM changes its state.

The APM is based on the information about dependencies and correlations between the requests of different services, as well as the correlations of the service requests and the external factors such as seasons, weather, etc. By *dependencies* we understand causal relations, e.g., one service sending a message upon which the execution of another service depends. As opposed to dependencies, *correlations* may be caused by external reasons: e.g. an increase in demands of the flight reservations at a travel agency may have a clear correlation with an increase in demands of the hotel reservations.

An important source of information about the dependencies and correlations are execution *logs* registering events happening in the system, their originators, timing aspects, data and resources involved. Data mining offers a number of ready-to-use solutions for the correlation detection between the types of service requests and external factors. Moreover, execution logs were already successfully used in Process Mining [1] to (re)construct process models. In this paper we present techniques for the discovery of correlations between service requests using elements of both data mining and process mining. As we already mentioned, the SOA environment is a highly changeable one. Therefore, for some correlations not the whole log can be used as a source of relevant information. The research question here is the diagnostics which parts of the log should be considered as relevant.

The APM is based on the observations from the past. In the future we might observe serious deviations of the system behaviour from the behaviour predicted by the APM in the current mode. Therefore, once such a discrepancy has been established, the APM is adapted appropriately.

The remainder of the paper is organized as follows. In Section 2 we describe the Adaptive Predictive Model (APM). Initial construction of the APM is discussed in Section 3, its monitoring and adaptation in Section 4. Finally, we review the related work in Section 5 and conclude the paper by discussing its contributions and identifying directions of the future work in Section 6.

2 Adaptive Predictive Model

The core of our approach is the Adaptive Predictive Model (APM) capturing the expected system behaviour, expressed in the terms of correlations and dependencies. Every state of the APM reflects the characteristics of the system behaviour in some work mode. By work mode we understand a specific combination of *external parameters*, such as weather conditions or changes in the stock exchange indices, and service request rates for *interface services*, i.e., services that can be invoked by the environment. Non-interface services will be called *internal*.

Service request rates are described by their probability distribution parameters. In practical cases, we usually will assume service request rates to be distributed exponentially. The distribution parameter λ, in practice, will be estimated based on the observations of the service request rates. Formally, we introduce the following definitions.

Definition 1. *Let E be a set of external parameters with associated domains D_e, $e \in E$. Function $\upsilon : E \to \cup_{e \in E} D_e$ is called a* valuation *function if $\upsilon(e) \in D_e$ for all $e \in E$. Let S be a set of services and $S_0 \subseteq S$ be a set of interface services. With each service $s \in S$ we associate the probability distribution of its request rates $\nu(s)$.*

A work mode *w is a pair recording a vector of values of the external parameters and frequencies of interface services, i.e.,*

$$\langle \prod_{e \in E} \upsilon(e), \prod_{s \in S_0} \nu(s) \rangle.$$

The class of all possible working modes for given E and S_0 is denoted W_{E,S_0}.

While clearly, a system can reside in infinitely many work modes, we opt for a finite representation of the modes by APM states.

Definition 2. Adaptive predictive model *M is a pair $\langle \Sigma, \varphi, \rho, \delta \rangle$ such that Σ is a finite set containing states of the APM, $\varphi : \Sigma \to 2^{W_{E,S_0}}$ mapping states to sets of vectors of values of external parameters and distributions of the interface services' request rates, $\rho : \Sigma \to 2^{(S \times S)}$ mapping states to sets of pairs of significantly strongly (dis)agreeing (with each other) services, $\delta : \Sigma \to 2^{(S \times S)}$ mapping states to sets of pairs of dependent services.*

Recall that a *disagreement* (also known as negative correlation) indicates that the increase of the value of variable x corresponds to a decrease of the value of variable y, and vice versa. If the relationship between x and y is close to a decreasing linear relationship, i.e., to the relationship that can be described as $ax + by + c = 0$ with $a > 0, b > 0$, the correlation coefficients such as the Pearson correlation coefficient r [18] or Kendall's τ [24] will be close to -1. In the opposite situation, when the increase of x corresponds to the increase of y we talk about *agreement* (positive correlation). Should the relation between two variables x and y be close to an increasing linear relationship, i.e., to $ax + by + c = 0$ with $a < 0, b > 0$, the correlation coefficients are close to 1. If the correlation coefficient is close to 1 (-1) we say that an agreement (a disagreement) is *strong*; if the correlation coefficient is close to 0 we say that an agreement (a disagreement) is *weak*. Furthermore, we say that an agreement (a disagreement) is *significant* if the corresponding p value is small, i.e., it is unlikely that the relation has been observed just by chance. Important agreements and disagreements should be both strong and significant.

The APM is called *predictive* since we will predict the request rates for internal services (dependent variables) based on the dependencies and correlations as well as the interface services' request rates (independent variables). Moreover, in the presence of correlations between the interface services' request rates, knowing the request rates for *some* interface services we predict the request rates for additional related service request rates. Techniques that can be applied to this end belong to the well-studied research domain of regression analysis [17], and we do not elaborate on them further in the current study. The second adjective, *adaptive* is discussed in Section 4.

We say that an APM M is *complete* with respect to E and S_0 if for any $w \in W_{E,S_0}$ there exists $\sigma \in \Sigma$ such that $w \in \varphi(\sigma)$.

To illustrate the notion of APM consider the following example.

Example 1. A travel agency (cf. [26]) sells flights, stays at hotels and combinations (e.g., flight and hotel). Typical services involved would pertain to booking flights and hotels:

$$S = \{\text{“book a flight”}, \text{“book a conference room”}, \\ \text{“book a hotel room”}, \text{“return a booking confirmation”}\}.$$

These services constitute the set S. Interface services pertain to booking flights and rooms, i.e.,

$$S_0 = \{\text{“book a flight”}, \text{“book a conference room”}, \text{“book a hotel room”}\}.$$

Assume also that $E = \{\text{“period”}, \text{“daytime”}\}$, $D_{\text{“period”}} = \{\text{“regular”}, \text{“holiday”}\}$, and $D_{\text{“daytime”}} = \{\text{“working hours”}, \text{“outside the working hours”}\}$.

Let the APM M then be composed from $\{\sigma_{rw}, \sigma_{hw}, \sigma_{ro}, \sigma_{ho}\}$ representing working hours during a regular period, working hours during a holiday period, time outside the working hours during a regular period and time outside the working hours during a holiday period, respectively. We define $\varphi(\sigma_{rw})$ as $\{\langle \text{“regular”}, \text{“working hours”}, \varphi_{rwf}, \varphi_{rwc}, \varphi_{rwh}\rangle\}$, where the distribution φ_{rwf} is the $\nu(\text{“book a flight”})$ restricted to the observations corresponding to the working hours during a regular period, φ_{rwc} is the $\nu(\text{“book a conference room”})$ restricted to the same period and daytime and φ_{rwh} is the $\nu(\text{“book a hotel room”})$ restricted to the same period and daytime. Mapping φ for three remaining states can be defined in a similar way.

Correlations between services are expressed by the function ρ. For all states $\sigma \in \Sigma$ we expect $\langle \text{“book a flight”}, \text{“book a hotel room”}\rangle \in \rho(\sigma)$ reflecting the correlation between bookings of flights and hotels: usually, if a flight is booked, a hotel should be booked as well. Some correlations, however, hold only in certain states: for instance, business meetings are usually booked during the working hours of a regular period. Such a business meeting typically requires a conference room and a number of flights booked for the participants. Therefore, $\langle \text{“book a flight”}, \text{“book a conference room”}\rangle \in \rho(\sigma_{rw})$.

Finally, dependencies we consider in this example reflect the message exchange between the services. In our example, the message exchange occurs between bookings and confirmations and is independent from the APM state, i.e.,

$$\delta(\sigma) = \{\langle \text{“book a flight”}, \text{“return a booking confirmation”}\rangle, \\ \langle \text{“book a hotel room”}, \text{“return a booking confirmation”}\rangle, \\ \langle \text{“book a conference room”}, \text{“return a booking confirmation”}\rangle\}$$

for all $\sigma \in \Sigma$. □

We stress that the detection of dependencies and correlations as well as the distinction between dependencies and correlations are essential for efficient resource allocation: to improve the performance by reducing the data transfer time

between services. We might like to allocate actively communicating services to the same resource. On the other hand, correlated but independent services can be allocated to different resources to achieve a balanced spreading of the workload.

3 Initial Construction of the APM

In this section we discuss practical aspects of the initial APM construction. We postpone the discussion of monitoring and adaptation to Section 4.

One of the important sources of information about what services are currently active are execution logs registering events happening in the system, their originators, timing aspects, data and resources involved. Execution logs will be used to determine correlations. Detecting correlations is, however, challenged by the multiplicity of services and multiplicity of changes in the ongoing processes. Indeed, as indicated by industry, 50–100 services is named as a typical number of services being deployed at a company. According a Gartner study small companies deploy about 25 services on average while very large enterprises have a total amount of more than 1000 services [23]. Therefore, considering pairwise correlations of *all* services with *all* services, or of *all* services with *all* external parameters is not practical and we should develop a more scalable approach, e.g., restricting our attention only to a subset of services (Section 3.1). Once this question has been addressed, we proceed with discussing how the APM states should be defined (Section 3.2) and how correlations and dependencies should be determined (Section 3.3). To detect dependencies we supplement the information from the logs with the specification of service orchestrations. The presence of dependencies, such as message exchange, can be detected by observing the execution log. We should, however, be able to distinguish between message exchange corresponding to different orchestrations of services. To this end, we need to analyze the orchestration specification.

3.1 Determining "Important" Services

As mentioned above, studying pairwise correlations between all possible services and external parameters is not practically feasible. To understand how one can address this problem, recall Example 1. In this example ρ was in fact related to S_0 rather than to S. This should not be surprising as correlations between the interface services and correlations between the interface services act as a kind of "contract" between the environment and the SOA. Therefore, the validity of the APM is determined by the validity of its "contract" with the environment of the SOA, i.e., we should first consider correlations and dependencies between the interface services.

In order to make predictions, we need also to consider relations between the interface services and the internal services. The internal services can be invoked only by another service of the system, external or internal, i.e., every internal service is directly or indirectly invoked by an interface one. Therefore, we proceed

with studying correlations and dependencies between the interface services and the internal services.

From the practical point of view, interface services can be distinguished from the internal ones by analyzing orchestrations.

3.2 Defining States of the APM

Once the important services have been identified we need to define states of the APM. Recall that states depend on the values of external parameters and on the service request rates of the interface services.

External parameters While in Example 1 all domains of the external parameters were finite, this is not necessarily the case when continuous external parameters are considered, such as temperature or the Dow Jones index. Moreover, even if all domains of the external parameters are finite, considering all possible combinations of the external parameters might be impractical. Hence, we need to identify finitely many groups of "related" values of external parameters. This problem is a well-known clustering problem, common in statistical data analysis. Clusters obtained by means of one of the existing clustering techniques [16, 5] serve as the first candidate for the set of APM states.

Interface services Next we refine the set of clusters obtained so far by considering interface services. We would like to describe the service request rate of a given interface service over a given cluster by means of a well-known probability distribution. This is, however, not necessarily the case in practice as the actual distribution might have been different during different periods of time. In the travel agency example, for instance, the log might have recorded information pertaining to two different economic situations. If the economy is doing well, people take more flights during the holiday period than during the holiday period in crisis times. Failure to distinguish between the two economic situations would result in an ill-fitted statistical model. Therefore, we need to be able to distinguish between the event log parts corresponding to different situations. As we cannot *a priori* guess which external parameters might become relevant and explicitly take them into account in the APM, a different technique is required.

For each cluster C we need to carry out change detection analysis on the service request rates obtained for service s. To this end various nonparametric change detection techniques can be applied [10, 34]. Let $b_1, \ldots, b_n$ be the set of change points detected in the observed service request rates of s corresponding to C, and let $\nu^{C,s} = \{\nu_0, \ldots \nu_{n+1}\}$ be distributions corresponding to the observations preceding b_1, between b_1 and b_2, ..., after b_n, respectively. If no change has been observed we consider the distribution corresponding to the entire set of observations. Our first candidate to define Σ is hence $\Sigma_0 = \{\langle C, s, \nu\rangle | C \in \mathit{Clusters}, s \in S_0, \nu \in \nu^{C,s}\}$.

Example 2. Example 1, continued. Since there are only four possible combinations for values of external parameters we do not consider perform clustering.

Let the request rates for "book a flight" as observed during the working hours in a number of weeks of the regular period be as follows: 324, 287, 273, 313, 298, 215, 243, 237, 256, 221, 248, 296, 308, 284, 312, 288, 302. To estimate the change points in these time series, we follow [6, 34] and minimize the residual sum of squares of the following linear regression equation: $f_{i,j} = \beta_i + u_{i,j}$, where $f_{i,j}$ is the j's observation (request rate) on the segment i, β_i is a segment-dependent coefficient and $u_{i,j}$ is the residual corresponding to $f_{i,j}$. Applying this technique to the request rates above, we detect a change at the fifths and at the eleventh weeks of the observations. Hence, the partial samples we need to consider are $v_1 = (324, 287, 273, 313, 298)$, $v_2 = (298, 215, 243, 237, 256, 221, 248)$ and $v_3 = (248, 296, 308, 284, 312, 288, 302)$. □

Unfortunately, Σ_0 might be too fine grained: each distribution is determined based on a restricted number of observations, and hence, coefficients obtained by fitting models might be not statistically significant. Therefore, we try to make the sets of observations larger by joining different sets of observations. However, the sets joined should not be arbitrary: it should be likely that the observations originate from the same distribution. In practice various statistical tests can be used to determine whether two samples of observations originate from the same distribution. Since we cannot assume probability distributions to be normal, non-parametric tests should be used, e.g., the Mann-Whitney test, also known as Wilcoxon test, or Kolmogorov-Smirnov two-sample test [13]. In the presence of ties Kolmogorov-Smirnov test should be preferred.

Mann-Whitney test starts by computing the test statistic together with the corresponding p value. Depending on test statistic the null hypothesis (two samples of observations originate from the same distribution) may be rejected. To this end the test statistic is compared with two critical values (determined by a level of significance): if the statistic is less, than the smaller of the critical values or is greater than the bigger of the two, the null hypothesis is rejected. Hence, to obtain Σ for any pair of samples in Σ_0 we apply either the Mann-Whitney test or the Kolmogorov-Smirnov test and depending on whether the null hypothesis has been rejected, either join the samples or not.

Example 3. Example 2, continued. Using the Mann-Whitney test as implemented in R [29] we compute the the test statistic for v_1 and v_2 to be 32.5 (p-value = 0.01833). The critical values for 5% are 7 and 28. Since $32.5 > 28$ we reject the null hypothesis, i.e., v_1 and v_2 should not be joined. For v_1 and v_3 the test statistics is 21 (p-value = 0.6389) and the critical values for 5% are the same as for v_1 and v_3, i.e., 7 and 28. Since $7 \leq 21 \leq 28$ we cannot reject the null hypothesis and join v_1 and v_3. □

Note further that the set of states obtained as described above is not necessarily complete: certain combinations of values of the external parameters or service frequency rates can fail to be appear in the log. This set of states is, however, complete with respect to the log being considered as long as the clustering algorithm does not reject observations.

3.3 Correlations and Dependencies

Once the states of the APM have been determined, for each one of them we need to identify ρ and δ, i.e., the corresponding sets of pairs of services with strong (dis)agreement and message exchange. We start by considering ρ, i.e., correlations.

To identify the correlations holding in a state σ we first select all observations corresponding to σ. Next, given two services s_1 and s_2 and the service request rates of these services corresponding to σ we would like to use non-parametric correlation tests such as Spearman rank coefficient or Kendall's τ [13] to determine correlation between s_1 and s_2. Unfortunately, validity of this approach is undermined by the underlying assumption: the correlation between s_1 and s_2 in σ does not evolve with time. Hence, rather than considering the entire samples at once, we need to restrict our attention to a sliding window of samples similarly to change detection techniques [7, 11].

To detect the dependencies we need to analyse orchestrations. WS-BPEL, a popular language for specifying orchestrations [4], specifies the message exchange via partner links and distinguishes between static and dynamic processes. In static processes, the partner link information is defined at design time. In dynamic processes the partner link information is not known to the developer or needs to change at runtime to adapt to data or other dynamic requirements. To analyse dynamic processes we combine the log analysis with static analysis techniques originally designed for static processes.

Joined analysis of orchestrations and logs allows us furthermore to detect correlations between the services belonging to some long-running orchestration.

Example 4. Consider the following example. The tax office receives and proceeds tax declarations. Typical services would be "receive a declaration", "check a declaration", "send an invoice", "refund", "select a declaration for audit", "receive a complaint" and "process a complaint". Interface services include "receive a declaration", "select a declaration for audit" and "receive a complaint". While there is a correlation between the service request rates for "receive a declaration", "send an invoice" and "receive a complaint". This correlation refers to service rates with the shift of the sliding window: if, say, many declarations have been received in February, many invoices will be send in May and many complaints will be received in June. Combining the analysis of orchestrations with the analysis of logs allows detecting the size of the sliding window, and, hence, the prediction of the request rates for time-separated service requests. □

4 Monitoring and Adaptation

Once an APM has been constructed, it should be constantly reconsidered with respect to new observations. To this end we introduce a *monitor* (Figure 1). The monitor observes the execution log, the values of the external parameters and the current state σ of the APM. If the correlations in $\rho(\sigma)$ and dependencies

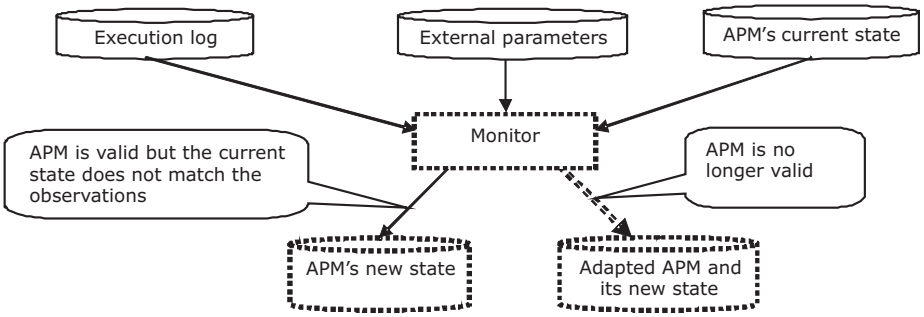

Fig. 1. APM monitor

in $\delta(\sigma)$ are still satisfied, we need to determine the state corresponding to the current work mode. If correlations or dependencies do not match $\rho(\sigma)$ or $\delta(\sigma)$ APM should be adapted. Since the sample observed might be too small and, e.g., not represent all possible states of the APM, we assume the correlations to hold (null hypothesis) unless the statistical analysis demands us to reject it.

Hence, for every sample of observations we need to determine the corresponding state of the APM and to check whether the correlations and the dependencies observed in the sample match the corresponding ρ and δ. To determine the state of the APM, we first determine to which cluster the combination of the values of the external parameters belong. Next we select the part of the sample corresponding to this cluster and analyse changes (cf. Section 3.2). Then we check whether the observations following the most recent change point and corresponding to the chosen cluster originate from the same distribution as the observations the APM has been based upon. If this is indeed the case, there exists a state, say σ, in the APM corresponding to the observations being analyzed. If this is not the case, the APM should be adapted by adding new states corresponding to possible combinations of the new distribution with the existing distributions and values of the external parameters in the same way we have constructed Σ_0 above.

Still, even if no new states should be added to the APM, it still might require an adaptation if, e.g., the sets of correlations and dependencies do not hold anymore. To decide whether ρ or δ should be adapted, we repeat the same process described in Section 3.3 and derive new values for ρ and δ.

5 Related Work

Importance of historical data has been commonly recognised. Histories [21, 22] and related notions such as event systems [31] and pomsets [19, 8] have been used in the past to provide a causality-preserving semantics for Petri nets. Unlike these works, we do not integrate historical data in the process models themselves but develop a model that is a *compressed* representation of the historical data.

Detecting "outdated" parts of the logs is related to the change-point detection [14, 12, 20, 33]: part of the log preceding the change-point should be disregarded. Unlike the approaches cited, we do not consider timed series, i.e., series of time-stamped events, but logs, i.e., series of series of time-stamped events. Moreover, no *a priori* knowledge of the distribution of the events is available. Still, as indicated above, we largely benefit from the existing work on nonparametric statistics [13, 10]. We apply the statistical techniques, however, in an entirely new setting and combine them with static analysis.

Logs were used in the past to derive information about the ongoing processes [1]. In this paper and the subsequent papers the notion of *process mining* has been introduced. Our approach is similar to process mining since in both cases execution logs are used to derive information about the ongoing processes. However, while process mining concentrates on deriving order-based process models describing the behaviour of *one process instance*, e.g., Petri nets [30], we focus on statistical models expressing correlations between services (activities) rather than on an order between them and consider a model of cumulative behaviour of multiple instances running in parallel.

Resource allocation for SOA has been considered, e.g., in [3, 32]. [32] considers prediction of workload dynamics of component services as requests traverse and pipeline through the workflow. The authors aim at predicting the incurred execution plan based on the given service transition probabilities, estimate the future workload of each component service, and finally allocate resources to them accordingly. The authors assume transition probabilities to be *given*, while we focus on *discovering* correlations, notion closely related to transition probabilities. [3] combines a short-term resource allocation with a long-term capacity planning problem. To address both problems the authors introduce the revenue/penalty system. Similarly to [32] the authors do not discuss prediction but consider existing workload forecasting methods [2].

6 Conclusions and Future Work

In this paper we have presented an approach that can be used as a preliminary step for adaptive resource allocation in SOA. The methodology focuses on the prediction of service request rates in service-oriented systems. We stress that understanding correlations and dependencies between the services is essential for efficient resource allocation. Therefore, correlations and dependencies form a core part of an adaptive predictive model (APM), upon which we will base our resource allocation. To construct the APM we combine static analysis of service orchestrations with statistical techniques, such as change detection, correlation testing and hypothesis testing.

While in this paper we focused on proposing a methodology, its experimental validation constitutes the main part of the future work considered. To this end we need to implement the approach and empirically compare different statistical techniques. Recall that we do not advocate the use of specific regression analysis techniques, correlation tests or change point detection methods but consider

them as "plug-ins" in our methodology. A separate study should be conducted to assess applicability and precision of different statistical techniques.

In addition to the APM we plan to develop the Data Exchange Model (DEM) estimating the volumes of data transferred between services. Since the data exchange rate is not reflected in the logs, it should be estimated by means of static analysis. As services can be implemented in a variety of programming languages, a generic-language based approach, such as ASF+SDF [9], is required. Note that we need to estimate the amount of information that will be exchanged by the services in the near future. As we do not know *a priori* the distribution of the amount of information being exchanged, we will develop a statistical model constructed by means of static analysis of the orchestration and implementations of services.

Finally, we aim at the integration of the prediction step described in this paper with existing resource allocation approaches such as [32]. Special attention will be paid to reducing the costs associated with the reallocation of the resources.

References

1. W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
2. B. Abraham and J. Ledolter. *Statistical Methods for Forecasting*. Wiley, Toronto, 1983.
3. J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian. Resource management in the autonomic service-oriented architecture. *Autonomic Computing, International Conference on*, 0:84–92, 2006.
4. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Technical report, BEA Systems, International Business Machines Corporation, Microsoft Corporation, 2003.
5. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999.
6. J. Bai and P. Perron. Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, 18(1):1–22, 2003.
7. P. Bauer and P. Hackl. The use of mosums for quality control. *Technometrics*, 20(1):431–436, 1978.
8. E. Best and R. R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.
9. M. van den Brand, A. van Deursen, J. Heering, H. A. de Jong, M. de Jonge, T. Kuipers, P. Klint, L. Moonen, P. A. Olivier, J. Scheerder, J. J. Vinju, E. Visser, and J. Visser. The ASF+SDF Meta-environment: A component-based language development environment. In R. Wilhelm, editor, *Compiler Construction*, volume 2027 of *Lecture Notes in Computer Science*, pages 365–370. Springer, 2001.

10. B. E. Brodsky and B. S. Darkhovsky. *Nonparametric Methods in Change-Point Problems*. Kluwer Academic, Dodrecht, 1993.
11. C.-S. J. Chu, K. Hornik, and C.-M. Kuan. The moving-estimates test for parameter stability. *Econometric Theory*, 11(4):699–720, August 1995.
12. C. Curry, R. L. Grossman, D. Locke, S. Vejcik, and J. Bugajski. Detecting changes in large data sets of payment card data: a case study. In P. Berkhin, R. Caruana, and X. Wu, editors, *Knowledge Discovery and Data Mining*, pages 1018–1022. ACM, 2007.
13. W. W. Daniel. *Applied Nonparametric Statistics*. PWS-KENT Publishing Company, Boston, 1990.
14. H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.
15. T. Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.
16. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, pages 226–231, 1996.
17. R. J. Freund, W. J. Wilson, and P. Sa. *Regression analysis: statistical modeling of a response variable*. Academic Press, Amsterdam, 2006.
18. F. Galton. Co-relations and their measurement, chiefly from anthropological data. *Proceedings of the Royal Society of London*, 45:135–145, 1888.
19. U. Goltz and W. Reisig. The non-sequential behavior of Petri nets. *Information and Control*, 57(2/3):125–147, 1983.
20. V. Guralnik and J. Srivastava. Event detection from time series data. In *Knowledge Discovery and Data Mining*, pages 33–42, 1999.
21. K. M. van Hee, O. Oanea, A. Serebrenik, N. Sidorova, and M. Voorhoeve. History-based joins: Semantics, soundness and implementation. *Data & Knowledge Engineering*, 64(1):24–37, 2008.
22. K. M. van Hee, A. Serebrenik, N. Sidorova, and W. M. P. van der Aalst. History-dependent Petri nets. In J. Kleijn and A. Yakovlev, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2007*, volume 4546 of *Lecture Notes in Computer Science*. Springer, 2007.
23. M. Jaeger and G. Rojec-Goldmann. SENECA – simulation of algorithms for the selection of web services for compositions. In *Technologies for E-Services*, volume 3811 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2006.
24. M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, June 1938.
25. J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.
26. D. Martin, J. Domingue, M. L. Brodie, and F. Leymann. Semantic web services, part 1. *IEEE Intelligent Systems*, 22(5):12–17, 2007.
27. L. O'Brien, P. Brebner, and J. Gray. Business transformation to soa: aspects of the migration and performance and qos issues. In *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 35–40, New York, NY, USA, 2008. ACM.
28. M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.
29. R Development Team. R homepage, 2008. Available at `http://www.r-project.org/` Consulted on March 8, 2009.

30. J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik. Process discovery using integer linear programming. In K. M. van Hee and R. Valk, editors, *Petri Nets*, volume 5062 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2008.
31. G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.
32. B. Wu, C.-H. Chi, Z. Chen, M. Gu, and J. Sun. Workflow-based resource allocation to optimize overall performance of composite services. *Future Generation Comp. Syst.*, 25(3):199–212, 2009.
33. K. Yamanishi and J. i. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Knowledge Discovery and Data Mining*, pages 676–681. ACM, 2002.
34. A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. `strucchange`: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38, 2002.