MONITORING SERVICE COMPOSITIONS IN MODE4SLA: DESIGN OF VALIDATION

Lianne Bodenstaff*, Andreas Wombacher, Roel Wieringa
University of Twente, Enschede, The Netherlands
{*l.bodenstaff, a.wombacher, r.j.wieringa*}*@utwente.nl*

Michael C. Jaeger
Berlin University of Technology, Germany
mcj@cs.tu-berlin.de

Manfred Reichert
University of Ulm, Ulm, Germany
manfred.reichert@uni-ulm.de

Keywords: Service composition, monitoring, management, validation.

Abstract: In previous research we introduced the MoDe4SLA approach for monitoring service compositions. MoDe4SLA identifies complex dependencies between Service Level Agreements (SLAs) in a service composition. By explicating these dependencies, causes of SLA violations of a service might be explained by malfunctioning of the services it depends on. MoDe4SLA assists managers in identifying such causes. In this paper we discuss how to evaluate our approach concerning usefulness for the user as well as effectiveness for the business. Usefulness is evaluated by experts who are asked to manage simulated runs of service compositions using MoDe4SLA. Their opinion on the approach is an indicator for its usefulness. Effectiveness is evaluated by comparing runtime results of SLA management using MoDe4SLA with runtime results of unsupported management. Criteria for effectiveness are cost reduction and increase in customer satisfaction.

1 INTRODUCTION

Service monitoring and management are challenging research areas. On the one hand research exists on composing services such that an optimal service level can be offered to the customer (Zeng et al., 2004), (Cardoso et al., 2004). On the other hand, research exists on monitoring performance of each service in the composition (Tosic et al., 2005). However, to our knowledge there is no research on monitoring compositions which take dependencies between corresponding services into account. Consequently, it is difficult to determine *why* Service Level Agreements (SLAs) of composite services are violated. Often badly performing services on which the composition depends cause these violations. Managers face the challenge of identifying these problematic services. Therefore, support in this tedious task is much needed. Effectively managing compositions results in competitive service level offerings to customers with maximum profit for the business. Furthermore, insights on services run by other providers, helps managers to plan negotiation strategies concerning SLAs. With the MoDe4SLA approach we monitor dependencies between services and their impact on the composition (Bodenstaff et al., 2008). However, so far, we have not evaluated MoDe4SLA. In this paper, we describe both our evaluation approach and evaluation criteria.

We aim at evaluating two aspects of MoDe4SLA. On the one hand we plan to evaluate how *useful* the approach is for managers burdened with actual maintenance of compositions. On the other hand we want to evaluate how *effective* MoDe4SLA is when maintenance is done using this approach. Since we do not have access to real composition monitoring data, we implement a simulator for running composite services. Experts from both industry and academia are asked to manage these generated compositions both with and without using MoDe4SLA.

To evaluate usefulness, we plan to interview experts, asking them to make a statement on how useful they perceived the approach when managing the compositions. To evaluate effectiveness we plan to test performance of compositions managed by experts using MoDe4SLA and of compositions managed by the control group not using MoDe4SLA. The level of *customer satisfaction* and *profits* made by the company

*This research has been supported by the Dutch Organization for Scientific Research (NWO) under contract number 612.063.409

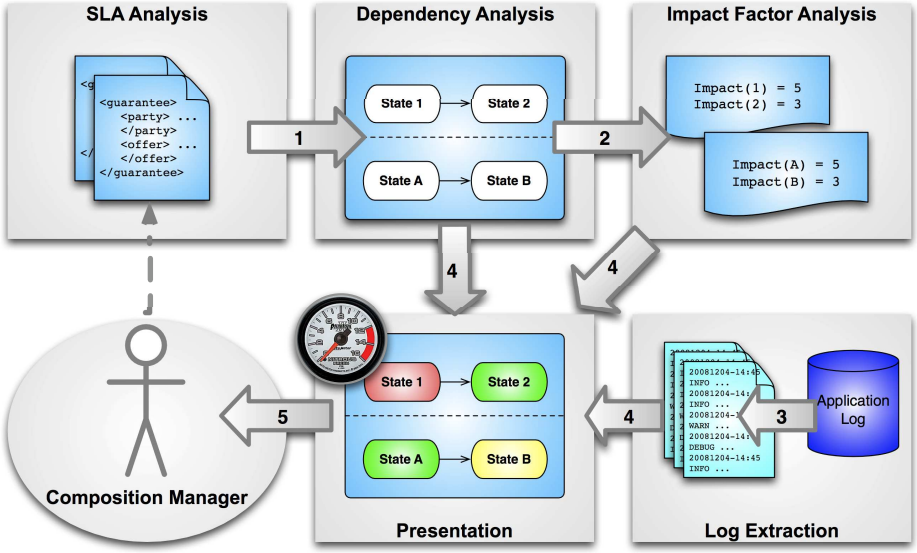

Figure 1: MoDe4SLA Approach.

are key criteria for evaluating effectiveness.

In Section 2 we discuss background needed for the understanding of the paper. We elaborate on the evaluation approach in Section 3, and conclude with related work and an outlook in Sections 4 and 5.

2 BACKGROUND: MODE4SLA

Companies offering composite services to their customers struggle to manage these complex constellations. Different services are provided with different levels of quality. These services stem from different providers, and have different levels of impact on the composition. Consider, for example, a service provider which allows financial institutes to check creditworthiness of potential customers. This service is composed of services querying databases, and a payment service where several payment options (e.g., PayPall and credit card) are provided. The company has an SLA with each of these service providers. The composite service offered to the financial institutes is dependent on all these services (e.g., payment services and database querying services). To meet the SLA with its customers the company faces the challenge of properly managing its underlying services, particularly since SLAs are often violated. For each of these violations the company has to determine how big their impact is on the composition, and it has to decide on which SLAs to renegotiate. Generally, complexity of this decision process grows with the number of services being involved in the composition.

Fig. 1 depicts the implementation of MoDe4SLA:

- At design time, we analyze the relations between different services and the composition with respect to the agreed *response time* and *costs* of the different providers (step 1 in Fig. 1).
- The result of this dependency analysis is the input for a subsequent impact analysis (step 2 in Fig. 1). The latter results in an impact model with intuitive graphical representation.
- During runtime, event logs are analyzed using the event log model, filtering events referring to the services, and their SLA statements for the composition (step 3 in Fig. 1). These results, together with the impact analysis and dependencies, constitute the input for the monitoring interface (step 4 in Fig. 1). The latter enables time efficient management and maintenance of the composition (step 5 in Fig. 1).

While we have described the MoDe4SLA framework in (Bodenstaff et al., 2008), this section provides a summarized description. Although MoDe4SLA is especially helpful in complex scenarios, we illustrate it by means of a simplified example. In this example a company offers composite service *NewsRequest* containing up-to-date news to its customers on a specific search query. Customers pay per invocation of the service. To gather the necessary information, the company invokes two content providers and sends information of the fastest responding one to the customer (cf. Fig. 4). Content provider 1 (CP1) is its main provider, offering fast response times (in 99.9%

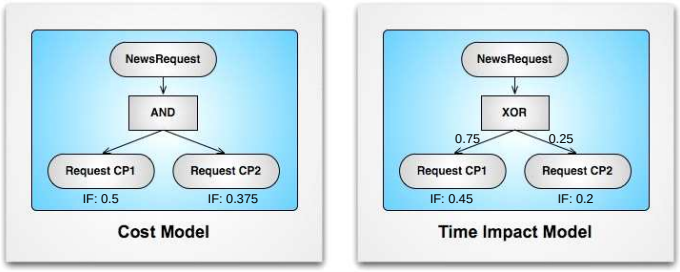

Figure 2: Impact Models.

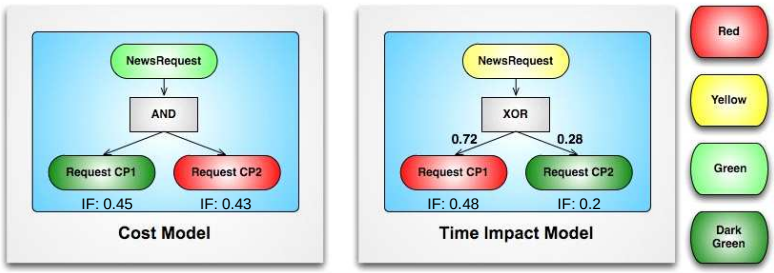

Figure 3: Feedback Model.

of the cases within 3 ms), accurate information, but for high costs (0.20 cents per invocation). The second content provider (CP2) constitutes the backup of the company in case the first one fails. This provider offers a slower response time and guarantees less accurate information but for a lower price.

Quality of service the company can offer to its customers is determined by the services it depends on. In our example (cf. Fig. 4) each service has an objective on *response time* as well as on *cost*. An *impact model* is derived for both objectives (cf. Fig. 2) which depicts dependencies between the different services for one run of the composition. Therefore, the impact models of the two objectives are different. E.g., when invoking both services in *NewsRequest* while using only data from the fastest responding one, response time depends solely on this service, while cost depends on both services. In addition an expected *impact factor* indicating the impact a service has on the composition is calculated for each service, per objective. E.g., CP1 in the time impact model is expected to be chosen 75% of the time (cf. Fig. 2), with an average response time of 3ms (cf. Fig. 4). The expected contribution per composition invocation for CP1 is therefore 3ms times 0.75 divided by 5ms, which is an impact factor of 0.45. Intuitively, services with a high impact factor require more attention when managing the composition than services with low impact.

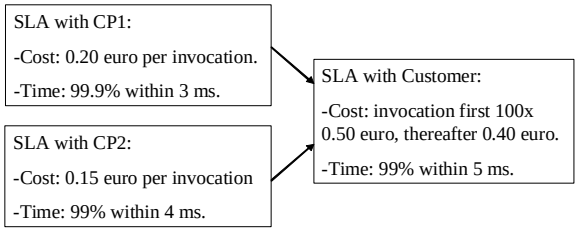

Figure 4: Service Composition: News Request.

At runtime the impact models are colored based on monitoring results (cf. Fig. 3) in comparison to the estimations made during design time. Thresholds for the different colors are set by the company. Fig. 3 depicts the graphical feedback the manager receives from MoDe4SLA. A red color indicates the service is performing worse than agreed upon in the SLA, while green indicates proper performance of the service. Yellow indicates the service is not performing perfectly but still within the boundaries set by the company, and dark green indicates a service running even better than the company anticipated. The edges are colored in the same manner, for example, red indicates the edge was chosen more often than expected. Fig. 3 indicates that the SLA with the customer is met regarding costs (i.e., it is colored green); by contrast the SLA with the customer is regularly violated regarding response time, although still within defined

boundaries (i.e., it is colored yellow). Bad performance concerning response time appears to be caused by content provider 1 (CP1), since its response times are significantly worse than agreed upon. The edges are colored for their slight deviation from the estimates made at design time. Also the realized impact factors are different from the estimated ones in Fig. 2.

3 VALIDATION APPROACH

To validate MoDe4SLA we need to determine how *useful* experts perceive the use of our feedback models (cf. Fig. 3). Furthermore we need to determine whether our approach is *effective*. The latter applies when it brings stakeholders closer to their goal, i.e., when the approach points to services causing bad performance of the composition. As basis for both approaches we extend and adopt SENECA (Jaeger and Rojec-Goldmann, 2005), a simulator for selecting services in a composition. In its current state, SENECA provides the means to generate random composition structures based on the workflow language patterns by (van der Aalst et al., 2003). In addition, the simulator also generates service candidates with randomly set QoS values for each candidate. E.g., considered QoS characteristics are response time, cost, and availability. In its first version, the simulator was designed to evaluate different selection algorithms for choosing the optimal set of service candidates for the composition. SENECA also implements QoS aggregation method for gaining overall QoS statements of the composition by aggregating the QoS of individual services (Jaeger et al., 2004).

3.1 Validation of Usefulness

To evaluate usefulness of our approach we use the following criterion:

> *MoDe4SLA is considered as being useful when experts testing it perceive the feedback given by MoDe4SLA as more useful for managing and maintaining the composition than when only using bilateral monitoring results.*

Common management approaches return bilateral monitoring results to the user. They do not provide information on the relation between the different services but merely return the performance of each individual service.

The evaluation setup consists of two parts. On the one hand we implement the composition simulator to generate, run, and analyze compositions. On the other hand we ask a group of experts to test the simulator and afterwards interview them to find out whether they perceive MoDe4SLA as useful.

3.1.1 Simulator

For evaluating the MoDe4SLA approach, the simulator is improved with the ability to perform a discrete event simulation when running the composition with service candidates. Fig. 5 depicts an overview of this simulator's functionality. SENECA randomly generates a *composition structure* for a given number of services. Actual considered structures in the simulator are sequences, loops, XOR-splits/XOR-joins, AND-splits and OR-splits (with either a normal join or a discriminative join). A discriminative join indicates that the structure succeeds when a subset of incoming services succeeded. For example, when only the fastest three out of five responding services are considered, then this is a discriminative join.

Each created service in the composition gets assigned a randomly generated SLA. The SLA of the composition is determined by the SLAs of the services it depends on. The impact models are derived using the structure of the composition as well as the agreed upon service levels. SENECA simulates the invocation of the candidate services according to the composition structure. Accordingly, services can fail with a particular probability, take longer, and, depending on their use in branching execution paths, different cost values will sum up. The simulator will generate log files reporting on the runtime results. In addition, the impact analysis as described in Section 2 is done and the impact models are generated.

3.1.2 Experts

The experts for our evaluation come from both academia and industry. We consider people to be experts if they are familiar with service compositions and SLAs. We start each session with a training period to explain our approach as well as the simulator. For the actual evaluation we prepare different types of compositions wrt complexity, i.e., how the structure is built, number of services, and diversity of SLAs. For each service monitoring results are gathered. Two documents for each composition are prepared. The MoDe4SLA document contains feedback (as shown in Fig. 3) while the control document contains the performance data for each service, but does not provide information on how they are related. Every expert is first offered the control document (cf. 1 in Fig. 5) and after that the MoDe4SLA document (cf. 2 in Fig. 5). We evaluate the following hypothesis:

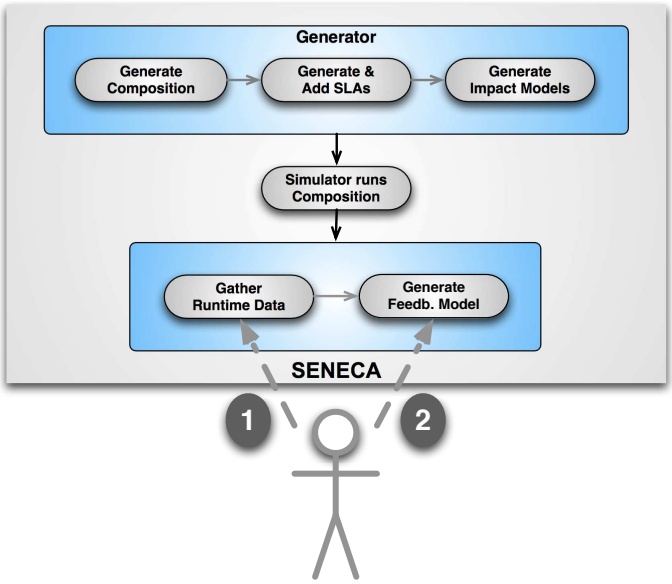

Figure 5: Evaluating Usefulness.

The MoDe4SLA document has a clear benefit over the control document for managing the composition.

For this purpose we will develop a questionnaire addressing several aspects. For example, does the expert save time by using MoDe4SLA, does he feel more confident in making decisions using MoDe4SLA, and does he find the approach helpful. A typical five-level Likert item is used to rate the response of the experts:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

3.2 Validation of Effectiveness

For evaluating effectiveness of our approach we use the following criterion:

The MoDe4SLA approach is considered to be effective if managing a service composition using this approach leads to better results than managing the composition without it.

We define *better* in this case as having better business results. In our opinion, this is a twofold process, where both *customer satisfaction* as well as *profit* are maximized (cf. Table 1). We assume customer satisfaction is high when the number of SLA violations is low, costs for the customer are low, and response time is low. We assume violations being associated with high penalties have bigger negative impact on customer satisfaction than low penalties. Therefore, customer satisfaction is measured through three criteria: payments for violations, average response time, and average costs for the composition (cf. Table 1).

	Minimize	**Maximize**
Customer Satisfaction	violation costs	
	average response time	
	average costs	
Profits	service costs	customer payments
	violation payments	violations paid by providers

Table 1: Effectiveness Criteria.

Maximizing profit is achieved through minimizing costs and maximizing income (cf. Table 1). In our case, these costs consist of costs for services and payments for penalties. Income is generated through payments by customers and payments by providers for SLA violations. The total payments for services to providers and for SLA violations paid to customers are subtracted from this income.

To evaluate effectiveness, we extend the implementation described in Section 3.1 so that it becomes possible to *rerun* the composition after some manage-

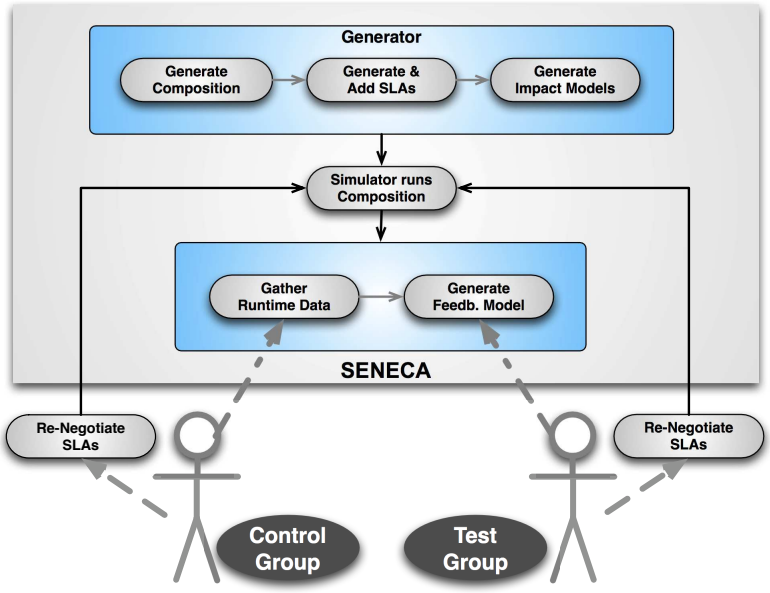

Figure 6: Evaluating Effectiveness.

ment choices have been made by the expert. Fig. 6 depicts an overview of the effectiveness evaluation approach. Furthermore, we adapt parts of the *service behavior*, we add a *renegotiation* of SLAs, and finally we *evaluate* effectiveness of the approach.

3.2.1 Service behavior

In real life the services in a composition behave differently. These differences arise, for example, because content of services is different, and they are offered by different providers. The difference in behavior, together with the structure of the composition, make it difficult to manage the services. In addition, managers can neither influence behavior of services provided by other companies nor can they predict their behavior other than relying on SLAs. To simulate this real life complexity, we implement *different types of behavior* for services. For example, some services have low variability in response time, and never violate the SLA while other services have higher variability in response time, and violate the SLA more frequently. Each service gets assigned a behavior type at design time. Which type of behavior is assigned to the service, is unknown to the expert.

To realize this, we implement *agents* to steer the different services in a composition. Each agent gets assigned a behavior type, for example, the *reliable* behavior type. Behavior of the service depends on both behavior type and agreed upon SLA. For example, if the behavior type is *reliable* and agreed upon response time is *lower than 10 ms*, a random distribution of response time will be generated for the next 100.000 invocations. This distribution is created with the parameters *reliable* and *lower than 10 ms*. At runtime, with each invocation of the service, a response time is randomly chosen from this distribution. As a result each service shows different behavior, fitting its SLA. Some services might violate their SLAs more severely, or more frequently than other services because of their assigned behavior type.

3.2.2 Renegotiation

After running the composition the expert is asked to renegotiate SLAs for a subset of the services. In practice, determining which SLAs to renegotiate is particulary difficult for complex compositions, i.e., compositions with many services and constructs. Since MoDe4SLA graphically pinpoints to badly performing services and, moreover, indicates the *impact* these services have on the composition, it becomes possible to prioritize SLAs, decreasing management efforts.

As described, services in the simulator expose unique behavior: some perform with only minor deviation in, for example, response time while others fluctuate more. In a perfect agreement, the SLA between provider and customer reflects this behavior exactly. In practice, however, it is necessary to monitor service behavior and renegotiate the SLAs for badly performing services. In these negotiations provider and customer have conflicting interest since both aim

at maximizing monetary benefits. To simulate this in the evaluation, the experts are asked to select a subset of services for renegotiation after the first run. Each service agent offers three new possible SLAs to the expert. E.g., the original SLA has fast response time but low violation penalties. A newly offered SLA by the agent might have slower response time, but high penalties when being violated. These new SLAs are generated taking the behavior type of the agent as well as the original SLA into account. Some of the new SLAs will fit better to the service from the customer perspective and some will fit the service better from a provider perspective. The challenge for the expert is to make choices beneficial for his company.

3.2.3 Evaluation

We use one *test group* and one *control group* for each tested service composition. The test group uses MoDe4SLA to choose services they want to renegotiate, while the control group uses log files only. MoDe4SLA is designed to assist the user in making choices on which services to renegotiate. Furthermore, MoDe4SLA pinpoints to the exact problems of the service so that choices between newly offered SLAs are easier to make. If the expert successfully identifies services with high impact on the composition and which are performing badly according to their SLAs, renegotiation will have to result in better runtime results of the test group than the control group (see criteria summarized in Table 1).

4 RELATED WORK

(Sahai et al., 2002) aim at automated SLA *monitoring* by specifying SLAs and not only considering provider side guarantees but focus also on distributed monitoring, taking the client side into account. (Barbon et al., 2006) enable run-time monitoring while separating the business logic from monitoring functionality. For each instance of a process a monitor is created. Unique in this approach is the ability to also monitor *classes* of instances, enabling abstraction from an instance level. The smart monitoring approach of (Baresi et al., 2004) implements the monitor itself as service. There are three types of monitors available for different aspects of the system. Their approach is developed to monitor specifically contracts with constraints. (Baresi and Guinea, 2005) presents an approach to dynamically monitor BPEL processes by adding monitoring rules to the different processes. These rules are executed during runtime. Our approach does not require modifications to the process descriptions what suits better to some application areas. An interesting approach in this direction is work by (Mahbub and Spanoudakis, 2005) which, as en exception, do consider the whole state of the system in their monitoring approach. They aim at monitoring derivations of behavior of the system. The requirements for monitoring are specified in event calculus and evaluated with run-time data. Although many of the above mentioned approaches do consider third parties and allow abstraction of results for composite services, none of them addresses how to create this abstraction in detail. E.g., matching messages from different processes as in our SubscribedNews example where databases are used, are not considered.

(Menasce, 2004) presents an analysis for the response time of composed services with the goal to identify the impact of slowed down services. The impact on the composition is computed by using a Markov chain model. The result is a measure for the overall slow down depending on the statistical likelihood of a service not delivering the expected response time. As opposed to our approach, Menasce's approach performs at design-time rather than providing an analysis based on monitoring runtime data. In addition, our work provides a framework a) to cover structures beyond an fork-join arrangement, and b) that supports different measures subject of an SLA in addition to response time. A different approach with the same goal is the virtual resource manager proposed by (Burchard et al., 2004). This resource management targets a grid environment where a calculation task is distributed among different grid nodes for individual computation jobs. The organization of the grid is hierarchically separated into different administrative domains of grid nodes. If one grid node fails to deliver the assured service level, a domain controller first reschedules the job onto a different node within the same domain. If this fails, the domain controller attempts to query other domain controllers for passing the computation job. Although the goal is the same and the approach covers runtime, it follows a hierarchical autonomic recovery mechanism. MoDe4SLA focusses on identifying causes for correction on the level of business operations rather than on autonomously performed job scheduling.

Another research community analyzes *root causes* in services. In corresponding approaches dependency models are used to identify causes of violations *within* a company. Here, composite services are not considered but merely services running in the company. E.g., (Agarwal et al., 2004) determine the root cause by using dependency graphs. Especially finding the cause of a problem when a service has an SLA with different metrics is here a challenging topic.

Also (Caswell and Ramanathan, 2000) use dependency models for managing internet services, again, with focuss on finding internal causes for problems. MoDe4SLA identifies causes of violations in other services rather than internally. Furthermore, our dependencies between different services are on the same level of abstraction while in root cause analysis one service is evaluated on different levels of abstraction.

5 SUMMARY AND OUTLOOK

In this paper we describe our plan to evaluate MoDe4SLA which is intended to support the company in managing its composite services for detection and coverage of SLA violations. We propose to do a twofold evaluation where first usefulness of MoDe4SLA as perceived by experts is tested after which effectiveness is evaluated by actually managing a service composition using MoDe4SLA. Both steps are supported by SENECA, a simulator which is extended for this purpose.

Currently, we are in the process of evaluating the usefulness of our approach. In near future we plan to extend our current implementation to also evaluate effectiveness. Furthermore, we extend our dependency based approach with handling complex SLAs where also dependencies *between* and *within* different SLOs occur, e.g., SLAs where two different SLOs concern response time, or where one SLO concerns not only response time but also cost.

REFERENCES

Agarwal, M. K., Appleby, K., Gupta, M., Kar, G., Neogi, A., and Sailer, A. (2004). Problem determination using dependency graphs and run-time behavior models. In *DSOM*, volume 3278, pages 171–182. Springer.

Barbon, F., Traverso, P., Pistore, M., and Trainotti, M. (2006). Run-time monitoring of instances and classes of web service compositions. In *ICWS '06*, pages 63–71, Washington, DC, USA.

Baresi, L., Ghezzi, C., and Guinea, S. (2004). Smart monitors for composed services. In *ICSOC '04*, pages 193–202, New York, NY, USA.

Baresi, L. and Guinea, S. (2005). Towards dynamic monitoring of WS-BPEL processes. In *ICSOC 05'*, pages 269–282.

Bodenstaff, L., Wombacher, A., Reichert, M., and Jaeger, M. C. (2008). Monitoring dependencies for SLAs: The MoDe4SLA approach. In *SCC (1)*, pages 21–29.

Burchard, L.-O., Hovestadt, M., Kao, O., Keller, A., and Linnert, B. (2004). The virtual resource manager: an architecture for SLA-aware resource management. *Cluster Computing and the Grid*, pages 126–133.

Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308.

Caswell, D. and Ramanathan, S. (2000). Using service models for management of internet services. *IEEE Journal on Selected Areas in Communications*, 18(5):686–701.

Jaeger, M. C. and Rojec-Goldmann, G. (2005). SENECA - simulation of algorithms for the selection of web services for compositions. In *TES*, pages 84–97.

Jaeger, M. C., Rojec-Goldmann, G., and Mühl, G. (2004). QoS aggregation for service composition using workflow patterns. In *(EDOC'04)*, pages 149–159.

Mahbub, K. and Spanoudakis, G. (2005). Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. In *ICWS '05*, pages 257–265.

Menasce, D. A. (2004). Response-Time Analysis of Composite Web Services. In *IEEE Internet Computing*, pages 90–92. IEEE Press.

Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A. P. A., and Casati, F. (2002). Automated SLA monitoring for web services. In *DSOM '02*, pages 28–41, London, UK. Springer-Verlag.

Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., and Ma, W. (2005). Management applications of the Web Service Offerings Language (WSOL). *Information Systems*, 30(7):564–586.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327.