Distributed Contracting and Monitoring in the Internet of Services

Josef Spillner[1], Matthias Winkler[2], Sandro Reichert[1], Jorge Cardoso[2], and Alexander Schill[1]

[1] TU Dresden, Nöthnitzer Str. 46, 01187 Dresden, Germany
{josef.spillner,sandro.reichert,alexander.schill}@tu-dresden.de
[2] SAP Research CEC Dresden, Chemnitzer Str. 48, 01187 Dresden, Germany
{matthias.winkler,jorge.cardoso}@sap.com

Abstract. The recent approval of the EU Services Directive is fostering the *Internet of Services* (IoS) and will promote the emergence of marketplaces for business and real-world services. From a research perspective, the IoS will require a new bread of technological infrastructures to support the concepts of business service description, contract management from various perspectives, end-to-end marketplaces, and business monitoring.

The IoS is a vision referring to web service-based digital societies. When service hosting moves from best-effort provisioning to guaranteed service delivery, monitoring becomes a crucial point of proof for providers and consumers of such services. We present the uplifting of technical contract monitoring results to business effects based on the distributed service infrastructure developed in project THESEUS, use case TEXO.[1]

1 Introduction

The emergence of electronic marketplaces for services is driving the need to describe services, not only at the technical level, but also from business and operational perspectives. In this context, Service-oriented Architectures (SOA) and web services leverage the technical value of solutions in the areas of distributed systems, cross-enterprise integration, and enterprise architectures. While SOA and web services reside in an IT layer, organisations are requiring advertising and trading business services which reside in a business layer. Previous solutions for Service Level Agreement (SLA) negotiation and monitoring need to be adapted to provide suitable infrastructures for the monitoring of the business aspects.

The European directive on services in the internal market [1] will facilitate businesses to provide and use cross-border services in the EU. It will also strengthen the rights of consumers of services, for instance by enshrining the

[1] The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The authors take the responsibility for the contents.

T. Senivongse and R. Oliveira (Eds.): DAIS 2009, LNCS 5523, pp. 129–142, 2009.

right of non-discrimination and contract fulfilment protection. In business, a service is the non-material equivalent of a good. It is considered to be an activity which is intangible by nature and is provided by a service provider to a service consumer to create a value possibly for both parties.

Real world examples of domains with requirements to digitally describe and monitor business services and establish contracts include the software industry (e.g. SAP Business ByDesign Services and IBM Smart Market) and automobile industry (e.g. BMW Assist, and Mercedes-Benz TeleAid). In these use cases, providers as well as consumers face the problem of describing service offerings, which is of considerable importance since services are one of the least understood portions of the global economy [2,3].

This paper is structured as follows: in Sect. 2 we explain the advantages of the Universal Service Description Language (USDL) as our approach to describing business services, the creation of SLA templates from USDL service descriptions and the negotiation of SLAs. In Sect. 3 we present our monitoring architecture and illustrate different aspects of IoS monitoring. In Sect. 4 we show how monitoring data can be aggregated and used to evaluate SLAs. Finally, we describe how discovered problems can be handled in Sect. 5, followed by a summary of the novelties of our approach.

2 Descriptions of Services and Service Level Agreements

The description of services is a fundamental requirement for enabling offering, search and usage of services. SLAs are formal contracts between a service provider and consumer regulating the provisioning and consumption. In this section we argue for a need of suitable means for describing services and present USDL as our approach. We will also show how SLAs are created based on USDL descriptions.

2.1 Business Service Descriptions

Recently, the vision of the IoS [4] and service marketplaces have emerged and can be seen as a new business model that can radically change the way users discover and invoke services. The development of infrastructures to maintain electronic marketplaces for services will require the support for the contracting and monitoring of business aspects of services. In the IoS vision, services are seen as tradeable goods that can be offered on service marketplaces by their providers to make them available for consumers. Barros et al. [5] describe service marketplaces as one example of service ecosystems that represent "[...] a logical collection of [...] services whose exposure and access is subject to constraints, which are characteristic of business service delivery." On a service marketplace multiple providers may offer their business services, thus creating an ecosystem which enables competition as well as collaboration among service providers.

Going beyond WSDL. The notion of business service is broader than the well-known concept of web service. Web services have mainly an information

technology (IT) perspective. They are technical software resources which are discoverable, invokable, platform independent, and self descriptive [6]. This type of service is mainly described by an interface definition (e.g., WSDL and other WS-* protocols) with a focus on technical service aspects. SLAs and monitoring consider the technical and infrastructure level. The IoS has different requirements from the ones fulfilled with WSDL. While the technical description of services is important for SOA, the business and operational perspectives on services have a significant importance for the IoS. Therefore, new service descriptions are needed to bridge business, operational and technical perspectives. A suitable service description needs to account for information that includes legal constraints, pricing strategies [7], resources consumed and produced [8], service scope and purpose, consumer benefit, participating roles and responsibilities, service level, operations, distribution channels, and marketing endeavours. A better description of the business and operational perspectives will bring to a marketplace an advantage over competitive platforms by being an added value for service providers and consumers. Based on this examination and requirements, we have devised a new specification language - USDL: the Universal Service Description Language - for services that will be hosted and traded in electronic marketplaces.

Describing Services with USDL. The Universal Service Description Language [10] enables the description of business characteristics exposed by an organisation for the purpose of providing a way for consumers to invoke and use services. The USDL schema defines three core clusters of information: business, operational and technical. Fig. 1 shows a simplified view of the USDL meta model. It can be seen that USDL has a strong emphasis on business and operations, while the technical perspective is reduced. The business cluster is used to describe information about the service provider and relevant consumers it is destined for, quality of service aspects, legal information, and marketing information such as pricing. Also, interaction aspects regarding service invocation and execution and bundling information is described. The operational cluster describes the offered functions of a service and provides a functional classification which supports the search for a service. Finally, the technical perspective allows the specification of different WS-* protocols for interaction. By defining the three clusters USDL goes beyond purely technical approaches such as WSDL. On the other hand it provides a well-understood and limited set of options for describing the most important aspects of business services. This approach is different from e.g. ontological approaches such as WSMO [9] which enable the user to model complex descriptions, but have the drawback of being difficult to handle by business users. More details on USDL can be found in [10].

USDL: An example from logistics. Listing 1.1 presents a simplified example of a USDL description (business and operational aspects) of a logistics service. The example describes the Truck Transport service that enables the transport of goods within the city limits of Dresden. It is classified as a logistics service according to the UN/SPSC standard[2]. The service will be executed within

[2] United Nations Standard Products and Services Code, `http://www.unspsc.org/`

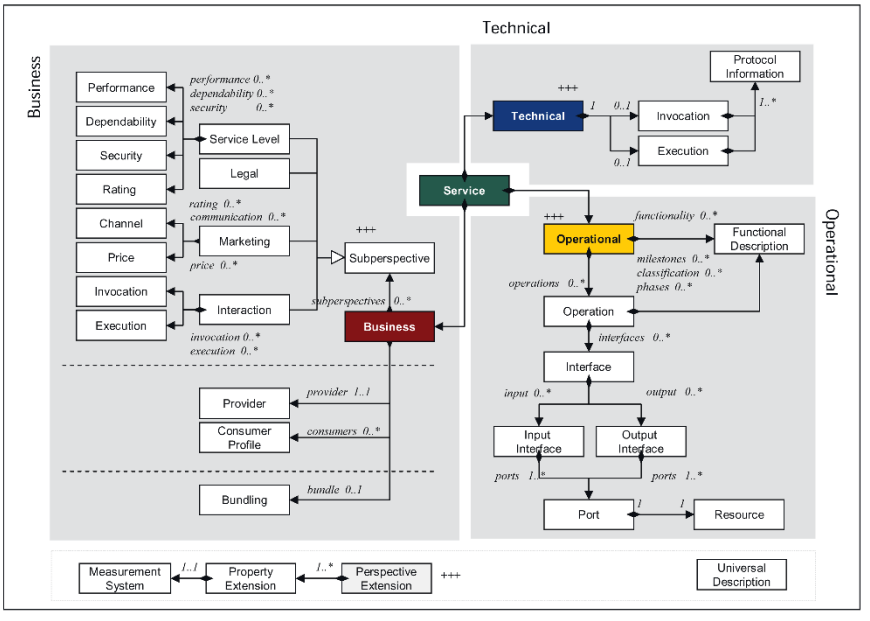

Fig. 1. Simplified view of the model behind USDL

3 hours and has an advertised reliability of 95%. This example will be used in the next section to exemplify how an SLA template can be generated automatically on behalf of the provider.

```
service {
  serviceName Truck Transport
  description Transport of goods within city area
  business {
    providerName Truck Transport Dresden GmbH
    providerAddress Traubestr 17, Dresden, Germany
    price 100 EUR
    termsOfUse http://www.truck-dd.com/ToU.html
    executionTime 3H
    reliability 95%
  }
  operational {
    classification_UNSPSC 80111623
}}
```

Listing 1.1. Sample USDL for logistics service

2.2 Deriving Service Level Agreements from Service Descriptions

In the IoS vision SLAs provide a formal base regulating the provisioning and consumption of services between service providers and consumers. These contracts are monitored to assure conformance to the agreement by both involved parties. Violations of the different service level objectives (SLOs) of an SLA need to be identified and reactions triggered.

Different technologies have been developed in recent years for negotiating and representing such formal contracts (e.g. WSLA [11], SLAng [12], WS-Agreement [13]). While WSLA and SLAng are not being developed any further, the WS-Agreement specification is driven by the Open Grid Forum. It provides a structure and language for specifying SLAs as well as a protocol for offering and negotiating SLAs. For our purpose we have chosen to implement SLA handling based on WS-Agreement and augment it with information from our USDL specification. The creation of SLAs is integrated with our service development process. A runtime component, called SLA Manager, encapsulates SLA syntax and handling. Its task is the negotiation of SLAs and making SLA information available to other components. In the following sections we will describe the extended WS-Agreement structure as well as the implementation of the SLA Manager.

Specifying Service Level Agreements. The SLA negotiation process, which follows the protocol specified by WS-Agreement, has an SLA template as its starting point. It is generated from the service description at the end of the service development process. During the negotiation process this template is refined first to an agreement proposal and finally to an agreement. The different WS-Agreement files are structured in mainly three sections: the *ServiceDescriptionTerms*, *ServiceProperties* and *GuaranteeTerms*. The *ServiceDescriptionTerms* section describes general information on the service and the functionality it provides including but not limited to the service name, pricing information, terms of use, and a functional classification. The *ServiceProperties* section defines measurable service attributes (e.g. execution time). The *GuaranteeTerms* section defines SLOs (e.g. min, max, average, or concrete values) which are guaranteed for service provisioning. They can be specified for the variables defined in the *ServiceProperties* section. A simplified SLA example is shown in Fig. 2. Due to the space limitations of the paper not all attributes are shown in the SLA.

In order to create SLA documents for services, a language for describing services is needed in addition to the language constructs of WS-Agreement. USDL provides such functionality. Thus, we have used it within SLA documents. Fig. 2 depicts examples of USDL code marked via the `usdl` namespace within an agreement.

Fig. 3 presents an overview of the SLA generation, negotiation, and monitoring processes which support tradeable services. In our implementation thereof, services are created using a service engineering workbench which is called ISE (Integrated Service Engineering). It implements a model-driven approach to service development and was developed based on the Eclipse platform. As a final step of the development process ISE generates SLA templates from the USDL service description. There are two ways for integrating information from a USDL

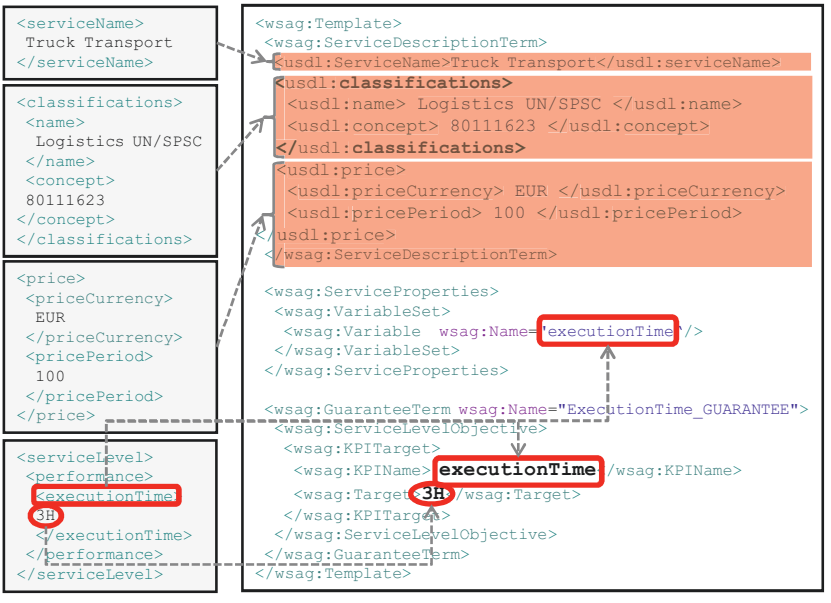

Fig. 2. Mapping USDL to a WSAG template

service description into an SLA template file. The first approach which we apply is to embed sections of USDL code into the WS-Agreement document structure. This is done in order to create the *ServiceDescriptionTerms* section, where a domain-specific service description language is needed. The second way of integrating service description information into the SLA is the mapping of USDL parameter names and values to WS-Agreement elements. This mapping is used for the generation of the *ServiceProperties* and *GuaranteeTerms* sections. Fig. 2 illustrates the mapping between USDL and a WS-Agreement template. Four simple USDL fragments (service name, classification, price, service level) are mapped to the different sections of the template. We implemented this transformation using openArchitectureWare [14]. The generated templates are then deployed to the SLA Manager where they are available for the negotiation process which is described in the next section. An approach of generating SLA templates from service descriptions was also described in [15]. It is limited to purely technical service aspects, while our approach, through the usage of USDL, allows to specify also business related service aspects such as rights and duties of the involved parties and penalties, to only mention a few.

The SLA Manager. The SLA Manager is a central component of the Service Management Platform (see Fig. 4), which handles a variety of tasks related to SLAs. First of all it provides interfaces for the deployment, update and removal of SLA templates. These interfaces are currently used e.g. by a deployment component which enables the deployment of newly modelled or changed services

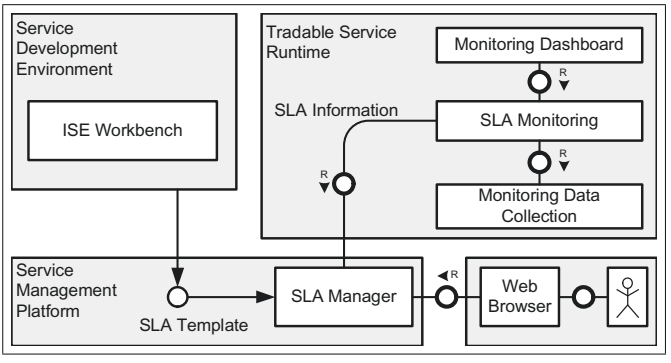

Fig. 3. Overview SLA generation, negotiation, monitoring

and related artefacts from the ISE workbench. The update functionality for SLA templates may also be used by other components, e.g. a monitoring component after realising that current SLAs are often violated and thus the SLA template needs to be refined accordingly.

The main task of the SLA Manager is to support the negotiation of SLAs which follows the approach defined by WS-Agreement. The negotiation is started by a user who intends to consume a service. The SLA Manager provides an SLA template which is presented to the consumer via a user interface as part of the Service Management Platform. It allows the consumers to make changes to the SLA template and submit it in the form of an agreement proposal. This document is validated by the service provider and accepted or rejected.

Another major task of the SLA Manager is the monitoring of the state of negotiated SLAs. While the evaluation of SLAs based on monitoring information is executed in a distributed fashion by the SLA Monitoring components at the Service Delivery Platforms (see Fig. 4), the SLA manager keeps track on SLO violation information from these components. It provides an interface for SMP components such as Billing to retrieve information regarding the state (SLA fulfilled, violated, not determined) of an SLA as well as the types of SLA violations that occurred.

Further interfaces are provided for information about contractual details which are needed by the SLA Monitoring components (SLOs to monitor) or by any subsequent pricing and billing components, e.g. for pricing information and consumer data.

3 Contract Monitoring

The task of contract monitoring is to collect all information necessary to realise the execution of tradeable services with respect to given guarantees (SLA) and to get usage data relevant for billing. On a technical level, service and system monitoring help reaching this goal. We present a monitoring architecture which integrates the flow of contracts.

3.1 Distributed Monitoring Architecture

In Sect. 1 we briefly introduced the proposed IoS architecture, consisting of one Service Management Platform (SMP) as central marketplace and several, distributed Tradeable Services Runtimes (TSR) for hosting the services. The consideration of all requirements produces the contracting and monitoring architecture illustrated[3] in Fig. 4. The main building blocks at TSR level are the Process and Service Engines, Access Gate, Adaptation Container and TSR Monitoring. At SMP level the blocks are SLA Manager, Monitoring Backend, Access Rights Management and components for further processing. The communication internal to TSR and between TSR and SMP is accomplished via a message-oriented middleware (MoM) to efficiently send events to multiple recipients.

When a new service is deployed, its code is transferred to the Process and Service Engines at the TSR. Once a customer has negotiated a contract via the SLA Manager's SLA Negotiation component, the resulting SLA is stored in the SLA Repository and the SLA Manager sends a message to the MoM that a new SLA is available. Subscribers of this type of message are SLA Monitoring and Monitoring Coordinator at TSR Monitoring. The latter then starts the appropriate Monitoring Sensors and Aggregators as described in the following sections. In case of an SLA violation, the SLA Monitoring triggers the Adaptation Coordinator to start one of the Adaptation Mechanisms described in Sect. 4.3.

Since complex business processes may consist of multiple services, deployed on distinct TSRs, a central Monitoring Backend at SMP level is needed to collect the monitoring data from single services and merge it into a central database. Consumers with further processing needs can access the monitoring data via Monitoring as a Service (MaaS). To keep SLA-related data private, MaaS checks the requester's identity and the access rights at every request.

In the following subsections we highlight the challenges of the various Monitoring Sensor types. We distinguish between monitoring on the system and execution container level (Sect. 3.2) on the one hand, and individual service monitoring on the other one. A further difference exists in that some of the service properties can be monitored from the outside (Sect. 3.3), e.g. by observing its message transmission behaviour, while other properties can only be measured with explicit support within the execution container (Sect. 3.4). Finally, the sensor data is converted to business objects (Sect. 3.5) and linked to contract objectives.

3.2 System Monitoring

IT system monitoring is a well-established activity ranging from single desktop computers to large data centres. Usually, the overall health status of distributed hardware and software is determined by measurement with agents, e.g. using SNMP or Nagios [16], and controlled from a central monitoring location. In scenarios of contract-bound service execution, determining the status and available resources of the execution servers is mandatory for creating realistic SLA offers.

[3] FMC-notation, see http://www.fmc-modeling.org

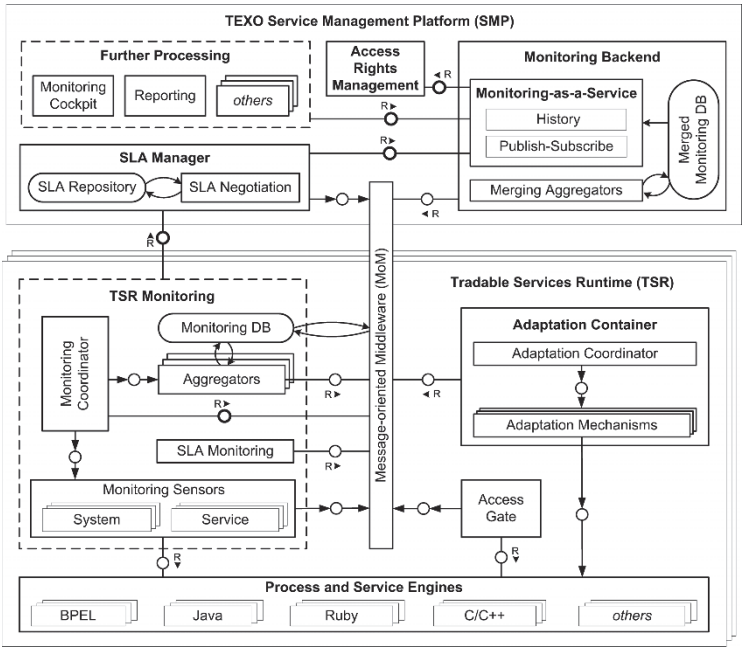

Fig. 4. Contracting and monitoring architecture for the IoS

In our approach, system monitoring controls the overall system health by keeping track of typical system parameters, e.g. system load, network performance, CPU and memory usage. For each SLO parameter found in active SLA files, the Monitoring Coordinator initialises a System Sensor which then transmits its measurements to the Monitoring DB, and the Aggregators for further processing, see Sect. 4.3.

In contrast to the system monitoring where only system-wide parameters are monitored, the following two categories covers all monitoring mechanisms which observe service specific parameters.

3.3 External Service Monitoring

External service monitoring mechanisms observe a service without the need for platform support. Parameters like the availability of a service can be probed by a third instance. Since these parameters are of a high importance to service providers, our monitoring framework possesses sensors and aggregators to monitor these non-functional properties of services.

An instance for external monitoring is the Access Gate. It represents a service by a transparent proxy which asynchronously intercepts all service invocations. In a first step, it checks the caller's identity by an authentication mechanism. In a second step, the Access Gate checks whether the caller is authorised to send this request. If positive, it forwards the intercepted message to the service

originally called, awaits the answer and sends it back to the originator of the request. The gathered usage information is sent to the MoM and will be used for billing purposes. Besides this, the Access Gate measures the response time, calculates the throughput of a call and sends the monitored value to the local Monitoring DB. If the caller can not be identified or is not allowed to send the particular request, an appropriate error message is sent to the MoM. The separation of concerns is maintained by encapsulating the authentication and monitoring code.

All of these monitoring operations are driven by SLAs which include both the objectives and the quality and therefore frequency of the monitoring probes. Due to often overlapping objectives, the probes are optimised by combining them.

3.4 Internal Service Monitoring

Going step by step closer from System Monitoring (see Sect. 3.2) to the services, parameters like CPU load or memory consumption are available at a more fine grained level for execution containers, e.g. a web server or the Java Virtual Machine, where all services share the same address space.

To gain even more knowledge about the status and behaviour of services, several techniques are available to inspect service instances at runtime. Most of them are based either on prior instrumentation, e.g. addition of monitoring status calls from within the service or opening up a shared memory structure to give insight into data structures, or on run-time instrumentation with tracing support from the execution environment (virtual machine, operating system). Tracing can be used to monitor the SLA compliance of a potentially untrusted service [17] whereas instrumentation is typically used for profiling and performance measurement. Either technique leverages the IoS concept of combining rapidly developed services with powerful execution platforms, leaving the measurement and management of services with specialised providers.

3.5 Business Monitoring

Based on the various available techniques for technical monitoring, higher-level business objectives in SLAs can also be monitored. Provider objectives like service popularity or increasing numbers of value contracts can easily be aggregated from existing sensor data. Consumer objectives like SLA compliance can likewise be controlled by using monitoring data. Therefore, we see the need to introduce aggregators and SLA checks on top of the already mentioned components.

4 Aggregation and SLA Status Determination

While the collection of monitoring data is a continuous process, a parallel activity to find out the interesting events and correlations is needed in order to determine the fulfilment of SLAs. We present an aggregation mechanism and an algorithm for SLA violation detection, and include methods to avoid SLA violations from happening at all.

4.1 Aggregation

On each service execution host, we assume the presence of one monitor. Sensors and aggregators run side-by-side as part of each monitor. While sensors collect data from various sources, aggregators turn such streams of data into higher-level indicators. To identify meaningful or complex events, reduce the amount of low-level events, and ensure the scalability of our system, we use existing complex event processing techniques. The uptime of a service is a good example for a non-measurable value which can only be calculated based on a series of individual test calls.

Since we assume a decentralised architecture with a central marketplace, another instance of the monitoring framework with special configuration runs on the marketplace. It only contains aggregators to further refine the results and produce cross-host metrics like the overall reliability of services available from that marketplace. In the previously introduced example of service guarantees in logistics, this can be seen in Fig. 5. Suppose that each incoming connection (1) gets redirected by a proxy to the service (2), while at the same time information about start and end times is measured (3) and broadcast across the monitoring infrastructure (4), (5). If the guaranteed response time of 3 hours is not met in at least 95% of all cases within a month, the aggregator sends an additional event (6) to the SLA Monitoring, which can then check the SLA violation status and transmit this information (7), (8) to the SMP to make it available to the user in a monthly report (9).

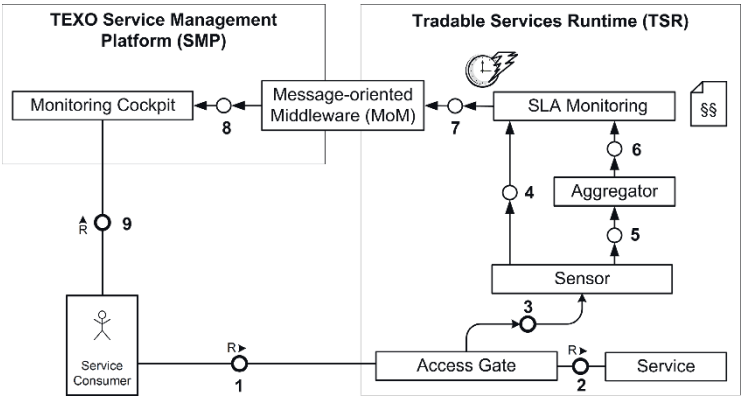

Fig. 5. Example of event propagation leading up to SLA violation

4.2 Determining SLA Conformance

We are currently developing a component for monitoring SLA conformance. Its task is to validate available monitoring information against negotiated SLAs. The SLA Monitoring component receives monitoring information via the MoM. Information on negotiated SLAs is requested from the SLA Manager. When the

violation of an SLO of an SLA is detected, an SLO violation message is sent to the MoM. From there the information is available to other components for triggering further actions (e.g. informing a responsible person) or displaying the information in the monitoring cockpit. An additional step following the monitoring could be the analysis of the effects of SLO violations. In service compositions, services are not isolated from each other. Instead, SLO violations of one service may lead to situations where other services cannot be provided any more. Monitoring such effects at runtime would help to improve the provisioning of services in compositions.

4.3 SLA Violation Prevention through Adaptation

Monitoring is not just an end in itself; rather, the collected and calculated data serves a very special purpose: to improve the quality of the service delivery. We distinguish between passive observation of monitoring data and active use for service adaptation, and argue for the necessity of adaptation to avoid contract violations.

Based on the information provided by the MaaS, the SLA Manager component decides if an SLA has been violated or is at risk of being violated in the near future as predicted by a probability-based forecast function. In such cases, adaptation can help avoiding the violation. Adaptation strategies include scaling-up by dynamically adding computing resources such as CPUs, memory or hard disk space, and scaling-down by reconfiguring the services or cutting down on some aspects of the contract. Adaptation mechanisms implement the strategies on a technical level by controlling certain targets like services or contracts. An Adaptation Coordinator (Fig. 4) is needed to prevent the collision and mutual neutralisation of the mechanisms. Upon completion of the chosen mechanisms, an adaptivity reasoner conveys this information into the service registry to adjust future contract template offers. We have based our categorisation of adaptation mechanisms on existing works, e.g. [18], but concentrated on a clear division between matchmaking time and runtime. The interplay between the coordinator, the reasoner, the mechanisms and the adaptation targets is shown in Fig. 6.

The effectiveness of adaptation shall be shown using the recurring example of a contract with a logistics service. In case an implied and agreed-upon tolerance region of a reliability of 95% is reached, e.g. at 96% after 50% of the associated time frame, the service can be reconfigured to increase the reliability at the expense of another property, most likely cost. This applies to both a technical sense of web service reliability and to a business sense of truck logistics reliability. In the given business-level example, assuming the main cause for belated transport is traffic congestion, the mechanism in question would modify the booking of trucks to insist on using faster, but more expensive, vehicle toll roads. Depending on the contract tariff scheme, this trade-off between toll and contract violation compensation can be an economic and reputation gain, as shown in Table 1.

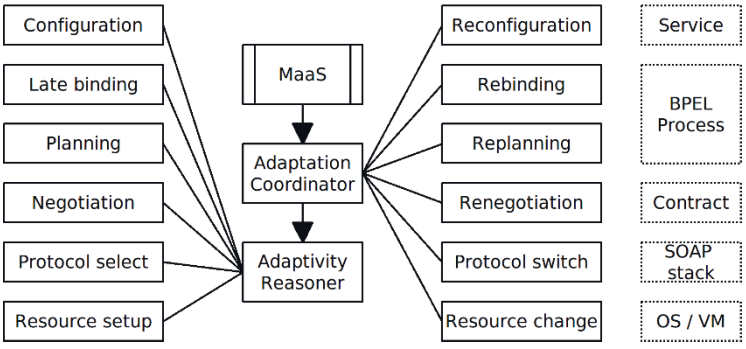

Fig. 6. Adaptation coordinator, reasoner, mechanisms and targets

Table 1. Cost-based adaptation trade-off

	Tariff without toll	Tariff with toll
Cost per transport	30 EUR	32 EUR
Probability of traffic congestion	7%	3%
Congestion compensation fee	50 EUR	
Resulting average cost	33.50 EUR	33.50 EUR
Effect on reputation	lowering	raising

5 Conclusion

We have designed and partially implemented a technical foundation for distributed service contracting and monitoring. A novel aspect of linking it to the business level was introduced. It allows consumers to rely on the advertised functionality of business services. The resulting architecture is built around USDL service descriptions and WS-Agreement based SLAs. Through a division into user-visible marketplaces and execution servers, it scales well enough for operation in an Internet of Services. The pervasive use of contracts and the enforcement of contractually guaranteed terms increases the acceptance among business users and makes it feasible to establish the excogitated service marketplaces.

Acknowledgements

The information in this document is proprietary to the following Theseus Texo consortium members: SAP AG and Technische Universität Dresden. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law.

References

1. European Parliament: EU Directive 2006/123/EC of the European Parliament and of the Council of 12 December 2006 on services in the internal market. Technical report, European Parliament (December 2006)
2. OECD: Business and Industry Policy Forum on the Services Economy. Technical report, Organisation for Economic Cooperation and Development (OECD) (2000)
3. Riddle, D.: Service-Led Growth. The Role of the Service Sector in World Development. Praeger Publishers, New York (1986)
4. Schroth, C., Janner, T.: Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IT Professional 9(3), 36–41 (2007)
5. Barros, A.P., Dumas, M.: The Rise of Web Service Ecosystems. IT Professional 8(5), 31–37 (2006)
6. Ameller, D., Franch, X.: Service-oriented computing: Concepts, characteristics and directions. In: WISE 2003: Proceedings of the Fourth International Conference on Web Information Systems Engineering, pp. 3–12. IEEE Computer Society Press, Washington (2003)
7. O'Sullivan, J., Edmond, D., Hofstede, A.: Formal description of non-functional service properties. Technical report, Queensland University of Technology (2005)
8. Dietrich, B.: Resource planning for business services. Commun. ACM 49(7), 62–64 (2006)
9. Roman, D., Lausen, H., Keller, U., de Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Oren, E., Polleres, A., Scicluna, J., Stollberg, M.: D2v1.3. Web Service Modeling Ontology (WSMO). WSMO Working Draft (October 2006)
10. Cardoso, J., Winkler, M., Voigt, K.: A Service Description Language for the Internet of Services. In: Proceedings of ISSS 2009 - International Symposium on Services Science (March 2009)
11. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. Technical report, IBM (2003)
12. Lamanna, D., Skene, J., Emmerich, W.: Specification Language for Service Level Agreements. EU IST 34069 deliverable D (2003)
13. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical report, Open Grid Forum (2007)
14. openArchitectureWare.org: openArchitectureWare. Project page
15. Reichert, J.: Serviceabhängige Qualitätsparameter in Dienstgüteverträgen. Java Spektrum (6), 29–33 (2008)
16. Toland, C., Meenan, C., Warnock, M., Nagy, P.: Proactively Monitoring Departmental Clinical IT Systems with an Open Source Availability System. Journal of Digital Imaging 20, 119–124 (2007)
17. Spillner, J.: Privacy-enhanced Service Execution. In: Westnik DUIKT - Proceedings of the International Conference for Modern Information and Telecommunication Technologies, Livadia, Krim, Ukraine (September 2008)
18. Meyer, H., Kuropka, D., Tröger, P.: ASG–Techniques of Adaptivity. In: Proceedings of Autonomous and Adaptive Web Systems, Dagstuhl, Germany (June 2007)