SLA-DRIVEN SERVICE MARKETPLACE MONITORING WITH GRAND SLAM

Josef Spillner, Jan Hoyer
Chair for Computer Networks, TU Dresden, Nöthnitzer Str. 46, Dresden, Germany
josef.spillner@tu-dresden.de, jan.hoyer@inf.tu-dresden.de

Keywords: Web services, SLA compliance, SOA

Abstract: Today, the number of functional equivalent web services increases rapidly. Service search engines and marketplaces are opening up to guide users to their preferred service providers. The question of what is the best service for the user's demand cannot be answered sufficiently. While semantic matchmaking techniques attempt to solve this question mainly for functional aspects, it is also important to know non-functional properties of a service like its availability and response time as well as the provider's level of fulfilment of contractual obligations regarding these properties. To realise the automated monitoring of services and contracts on service marketplaces, we introduce a standalone and easily integratable software component which is able to monitor non-functional properties and the adherence to negotiated contracts.

1 INTRODUCTION TO SERVICE MARKETPLACE MONITORING

Monitoring is an automated process of checking the condition of a system. For systems offering services to users, the term service monitoring refers to giving providers and consumers of these services the ability to check its behaviour and optionally match it against a set of requirements and expectations. If the requirements relate to participants who interact with the system, service-level agreements (SLA) are often used to manifest them on a contractual basis.

Web service and grid-based hosting environments with integrated monitoring and SLA management facilities are already widely available. However, when aiming at flexible, internet-scale web service hosting and trading, there are several aspects not yet covered by existing approaches. These include SLA-driven monitoring and aggregation as well as automated adjustments of service offerings based on non-functional property monitoring and prediction results. Grand SLAM has been designed to resolve these problems and be usable as a core component for researchers and builders of service marketplaces.

2 EXISTING MONITORING CONCEPTS AND PROJECTS

Conventional monitoring systems in production use perform server and service checks based on administrated configuration files. Sometimes, they allow service users to control the results of the checks. It is however not common among them to treat SLA as first-class objects. Nagios (Pervilä, 2007) is a widely used IT systems monitor which falls into this category and despite many extensions and a modular sensor design is not targeting service marketplaces. Keywatch (Keymind, 2008) is based on OSGi for modular deployment of sensors and filters, and offers a real-time query interface. SLA are similarly underrepresented in its design. Behavioural monitoring of BPEL processes is supported by Dynamo, which uses the WS-CoL rule language to define properties (Baresi et al., 2008). Rule languages are powerful but not sufficient to declare legally binding agreements.

Academic approaches have already acknowledged the weaknesses of conventional service monitoring and proposed SLA-centric designs. Fundamental issues related to SLA monitoring include a precise SLA specification language, flexible metric calculation, separation of the computation and communi-

cation infrastructure and violation detection services (Molina-Jimenez et al., 2004). Several projects exist to take these issues into account. SALMon is a system for monitoring services, analysing the results and taking decisions based on the outcome of the analysis (Ameller and Franch, 2008). It builds on the ISO/IEC 9126 quality model for non-functional properties, which is a very complete model but only contains a small subset of deterministically monitorable properties. Only two properties have been implemented into the monitor. The results are stored in a database, there is no interface for real-time notifications. Furthermore, SALMon does not seem to be publically available, making it difficult to assess the suitability of its internal design and architecture. Another project, SLAngMon, provides a monitor for SLA specified in the SLAng and WS-Agreement languages as part of the PLASTIC platform (Bertolino et al., 2009). It is based on a framework for online and offline testing of functional and non-functional service properties. Its tool PUPPET related to continuous monitoring on service marketplaces. However, the implementation only implements sensors for response time and reliability. The prototype is also lacking integration points for SLA management and access to monitoring data.

Particular challenges like SLA monitoring efficiency, accuracy and scalability have been analysed by theoretical work and validation tools, e.g. for network-related measurements (Sommers et al., 2007). Compared to this, only few integratable monitors exist. Due to our requirements on SLA-centric monitoring and integration, we have decided on providing an alternative framework which provides exactly these features and is aimed at being reusable by other projects by offering web service interfaces for SLA management and access to measurement data.

3 ARCHITECTURE OF GRAND SLAM

Grand SLAM is a modular system and web service monitor based on Java EE. The core of the monitor and the associated parts run on the OSGi Service Platform and use the OSGi framework for standardised module management. We decided for OSGi because it offers some advantages compared to an implementation in pure Java. On account of the OSGi Hot Deployment and the modular implementation of the framework, a higher fault tolerance is granted while starting, stopping, installing and updating measurement bundles. Defect bundles can be detected and stopped without terminating the whole monitor system. Therefore a higher stability of the monitor is granted. On the base of the service-oriented programming model of OSGi, it is possible to develop custom measurement bundles which are adapted to local server and web service specifications. For this matter, the framework provides a minimum of interfaces which have to be implemented. The monitor consists primarily of five parts as can be seen in Figure 1.

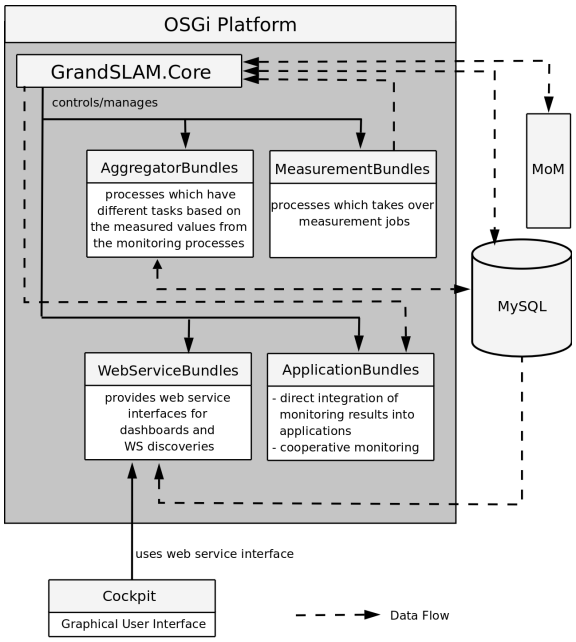

Figure 1: OSGi-based architecture of Grand SLAM

- A core monitor bundle manages and controls the other bundles and it is responsible for the supervision of the contracts which are stored in the local database. It installs a trigger on the database and is being notified of additions or removals which are caused by external SLA management and negotiation components attached to the Grand SLAM web service interface. The core bundle also stores raw data of the measurements and current states of the affected services into the database.
- Measurement bundles take over the measurement of one or multiple QoS parameters which are specified in the contracts and return these values to the core bundle.
- The third part of Grand SLAM are the aggregators. They are implemented as OSGi bundles and will be started by the monitor core but work independently from it. Resulting from this, there is no concrete default specification for their implementation, so it is possible that every aggregator

has a different task. In the current implementation of Grand SLAM there are three aggregators: One which generates aggregated values like average, minimum and maximum, a second one which creates scalar vector graphics which contain pie- and line charts and a last one which creates XML files with a ranking of the monitored services.

- The fourth part is an Axis 2 server provided by the open source OSGi Service Platform Knopflerfish. On this server, a web service is deployed which allows an invocation from a service discovery. Its task is to register and unregister service level agreements which have to be observed.
- The last part is optional and contains application-specific bundles for cooperative monitoring. Services which are aware of the presence of the monitor can adapt to its results explicitly. Apart from this integration, services monitoring is always enforced and adaptation must be triggered from external components.

Monitoring results are stored in a SQL database attached to the monitor via JDBC. Access to this data is provided by a SOAP query interface dubbed Monitoring-as-a-Service (MaaS). It offers convenience methods for retrieving historic data on services, contracts and providers, thus avoiding the need for direct access to the database. A cockpit with SVG charts has been created as a useful sample MaaS consumer. The MaaS interface also manages subscriptions to a message-oriented middleware for delivery of monitoring events in real-time. This communication channel is also usable to connect several instances of the monitor with each other. This way, a scalable hierarchical monitoring with a central service marketplace and decentralised service execution environments becomes possible. This aspect is not yet fully implemented but obviously needed for large service hosting sites. Service marketplaces can have many active contracts between various parties, as can be seen in Figure 2.

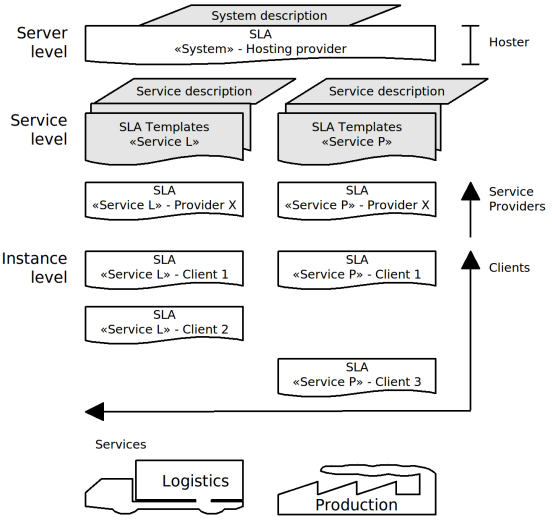

Figure 2: SLA between marketplace operator and service providers, consumers and server hosters

4 MONITORING OF SLA

Grand SLAM is able to observe Service Level Agreements (SLA) which are negotiated between service provider and service user. Based on our experience with evolving SLA languages, we have designed an SLA abstraction library which is able to extract objectives, expiration dates and other key data from both WS-Agreement and WSAG derivates.

An aspect central to all of these languages is the definition of measurement targets and their corresponding quality of service (QoS) parameters. A measurement target includes such information like service name, endpoint and server address. For each QoS parameter there also exist two ranges of values. The first one describes an allowed area for a measurement. If a measured value exceeds this area, the agreement is breached. The second one defines a critical area. A measured value will be marked as critical if the upper or lower threshold is exceeded. Another existing information is the scheduling parameter which defines how often a QoS parameter has to be measured. At the current implementation of Grand SLAM simple intervals in milliseconds can be defined. For the future we propose a complex scheduling mechanism where it is possible to set intervals in days or weeks or it would be possible to set concrete timestamps for when Grand SLAM has to measure the properties. This allows for defining the quality of the measurement process itself which could become part of the agreement pricing model.

Grand SLAM divides measurements into two groups. The first one, local monitoring, includes server specific measurements like the usage of a CPU. In this case only one instance of a measurement process exists to reduce the usage of resources allocated by Grand SLAM. The second group, remote measurements, are web service specific measurements like uptime or response time. These are individual QoS parameters of a service and, resulting from this, for every referenced parameter one instance of a measurement process exists.

The already existing measurement sensors and aggregation bundles are depicted in Figure 3. A par-

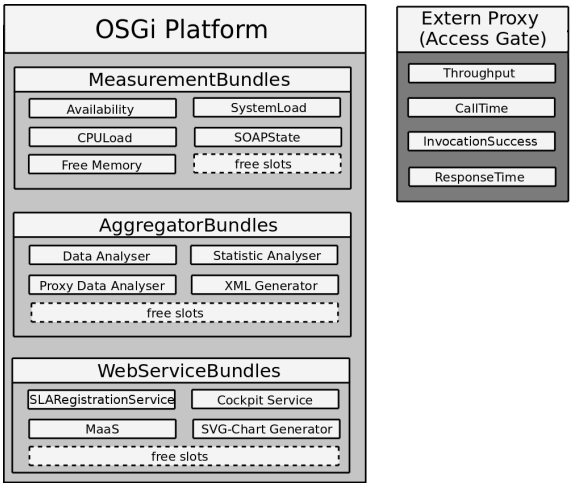

Figure 3: Measurement and aggregation bundles shipped with Grand SLAM

ticular design decision has been to integrate measurements from external system or marketplace components wherever they exist already. For example, the best place to measure service invocation response time and throughput for local services is a web service proxy. If no proxy is in use, the service SOAP stacks would have to be instrumented, which is not feasible for dynamically deployed (tradable) services. Similarly, operating system-level service execution properties like per-instance CPU usage can be injected by external tools for locally executed services.

Whenever a new contract becomes active, Grand SLAM parses the SLA and extracts the guaranteed parameters. Any measurement bundle not yet running will be activated at this point. Each bundle then provides a measurement task to the core bundle. Using an observer pattern, the core bundle will be notified of any new measurement as soon as it occurs. The core bundle is then able to evaluate the data, store it in its database and use a message-oriented middleware to forward it to a higher-level instance of Grand SLAM for further aggregation.

5 CONCLUSION AND FUTURE WORK

Grand SLAM has been introduced to perform SLA monitoring on service marketplaces. The collection and aggregation of monitoring information from the system, installed services and running contracts as well as the access to both real-time and historic information are essential features implemented by the prototype. Compared to existing projects, Grand SLAM offers good integration with usual service marketplace and SOA components like monitoring dashboards, service registries and execution adaptation. A number of service-related metrics can already be collected and reported with the included bundles. The addition of custom measurement and aggregation bundles is possible at runtime whenever required by additional negotiated SLA.

In the near future, it is planned to develop a mechanism which generates forecasting models from the collected measured values. Furthermore, the MaaS query interface will be extended to become more generic and efficient for a variety of monitoring data consumers. Finally, the already initiated implementation for distributed operation will be completed.

ACKNOWLEDGMENT

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.

REFERENCES

Ameller, D. and Franch, X. (2008). Service Level Agreement Monitor (SALMon). In *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS)*, volume 00, pages 224–227. Madrid, Spain.

Baresi, L., Guinea, S., Kazhamiakin, R., and Pistore, M. (2008). An Integrated Approach for the Run-Time Monitoring of BPEL Orchestrations. In *Towards a Service-Based Internet/ServiceWave*. Madrid, Spain.

Bertolino, A., Angelis, G. D., Frantzen, L., and Polini, A. (2009). The PLASTIC Framework and Tools for Testing Service-Oriented Applications. In *Software Engineering: International Summer Schools, ISSSE 2006-2008, Salerno, Italy, Revised Tutorial Lectures*, pages 106–139.

Keymind (2008). Keywatch - an open source, flexible, osgi-based monitoring system. http://www.keywatch.org.

Molina-Jimenez, C., Shrivastava, S., Crowcroft, J., and Gevros, P. (2004). On the Monitoring of Contractual Service Level Agreements. In *First IEEE International Workshop on Electronic Contracting (WEC)*, pages 1–8. San Diego, California, USA.

Pervilä, M. A. (2007). Using Nagios to monitor faults in a self-healing environment. Seminar on Self-Healing Systems, University of Helsinki.

Sommers, J., Barford, P., Duffield, N., and Ron, A. (2007). Accurate and Efficient SLA Compliance Monitoring. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 109–120. Kyoto, Japan.