X-FEE
An Extensible Framework for Providing Feedback IN the Internet of Services

Anja Strunk
Department of Computer Science, Institute of Systems Architecutre
Dresden University of Technology, 01069 Dresden, Germany

Keywords: Internet of Services, Service-oriented Architecture, Web Services, Feedback, Monitoring, User Feedback.

Abstract: In the Internet of Services, there is a big demand for feedback about services and their attributes. For example, on service market places, feedback is used by service discovery to help a service user to find the right service or at runtime, feedback is employed to detect and compensate errors. Thus, the research community suggests a large amount of techniques to make feedback available. However, there is a lack of adequate feedback frameworks to be used to implement these techniques. In this paper we suggest the feedback framework X-Fee, which is highly extensible, flexible and interoperable to easily realize feedback components and integrate them in arbitrary infrastructures in the Internet of Services.

1 INTRODUCTION

The vision of the future internet is to transform the web of information to a web of services. The goal is to develop a so-called *Internet of Services* (IoS) where economically viable services are offered, brokered and consumed via the internet. Service marketplaces emerge as web platforms where providers can publish and sell services and customers can find suitable services and combine them together according to their needs.

While trading and using services in the IoS, a lot of feedback information arises. The most important are monitoring and user feedback. (Kalepu et al, 2002).

Monitoring feedback results from quality monitoring. In the IoS, each service guarantees a specific QoS level, such as receiving the service's response within a certain time period. The assured QoS properties, negotiated between the service provider and the service user are called *Service Level Objective* (SLO) and saved in a legal contract, called Service Level Agreement (SLA). At runtime, quality monitoring observes the compliance of each SLO by comparing the measured values with the one that was promised. Thus monitoring feedback deduces the service's reputation related to their QoS and their SLO compliance.

User feedback is created by human beings, after each service interaction, by evaluating non-measurable service attributes, such as correctness or price-benefit-ratio. Hence it allows determining the service's reputation related to the non-technical aspects.

The research community has suggested different approaches to make both monitoring and user feedback available in the IoS. In fact, there is a great demand for feedback. For example, at design time, feedback is consumed in order to improve existing services or to deduce innovative ideas for new services (Stathel, et al, 2009). The service discovery, a search engine for services, integrates feedback to support the user by selecting the right service. (Xu et al., 2007). At runtime feedback is used to recognize errors and to trigger appropriate compensation mechanisms (Strunk et al., 2009).

Despite the great demand for feedback and a lot of techniques to provide it, there is a lack of an adequate framework, which can be used to implement these techniques. State-of-the-art feedback components are not modular, interoperable or flexible enough to be reused in new platforms. This forces a costly re-development of already existing software components.

In this paper we propose the feedback framework X-Fee, which is highly extensible, flexible and interoperable to be easily integrated in arbitrary infrastructures. It acts as a base for the implementation of methods for providing both monitoring and user feedback.

The rest of this paper is organized as follows. Section 2 presents an overview of related work in this area, followed by a description of the feedback model used in X-Fee. The overall architecture of X-Fee is explained in Section 4. In Section 5 we discuss the strength and weaknesses of our approach and finish the paper with a conclusion.

2 RELATED WORK

Feedback in service-oriented architectures and the IoS is very well investigated. Started with the question how feedback information can be formalized, Maximilien and Singh (Maximilien and Singh, 2002) published their conceptual model of web service reputation. Today, this feedback model is the de-facto standard and is also the basis for X-Fee.

Kalepu et al. analyse feedback in the IoS and categorize it into monitoring and user feedback (Kalepu et al, 2002). With the author's definition of reputation as f(User Ranking, Compliance, Verity)", they argue that both feedback categories have to be considered to get meaningful results.

The first approaches for techniques to provide monitoring feedback are presented by Robinson (Robinson, 2004) and Fickas et al. (Fickas et. el, 1995). Both authors propose a technique to monitor web service requirements. Raimondi et al. (Raimondi et al., 2008) investigate monitoring of SLOs more closely and define a methodology for online monitoring of web service SLOs based on timed automata.

Beside techniques to monitor SLOs, commercial and academic monitoring systems are developed. However, the main disadvantages of these approaches are the lack of interoperability, flexibility and extensibility.

The well-know commercial monitoring system is Nagios (Pervilä, 2007). Despite its many extensions and modular sensors, it is not flexible enough to monitor SLOs. Two meaningful academic monitor approaches are Grand Slam (Spillner et al., 2009) and SLAMon (Ameller et al., 2008). Both are modular systems, which can be extended by future functionality. Unfortunately the core modules can not be adapted. Thus the fields of usage of Grand Slam and SALMon are limited; for instance, the used SLA description format can not be changed. On the contrary, X-Fee allows modifying any part of the system.

In contrast to SLO-monitoring, techniques and systems related to user feedback are well analysed and discussed. One of the best survey papers in this area is the one by Wang and Vassileva (Wang and Vassileva, 2007). The authors compare more than ten academic as well as productive systems to collect, compute and provide user feedback. None of these approaches, however, is interoperable, flexible and extensible enough to be used in heterogeneous systems and to be adapted to individual requirements of system providers.

3 FEEDBACK MODEL

Feedback in the context of the IoS evaluates a service's functional and non-functional attributes. It is created for each service separately after each interaction.

X-Fee organizes the different kind of feedback information and the resulting reputations in a conceptual feedback model, depicted Figure 1.

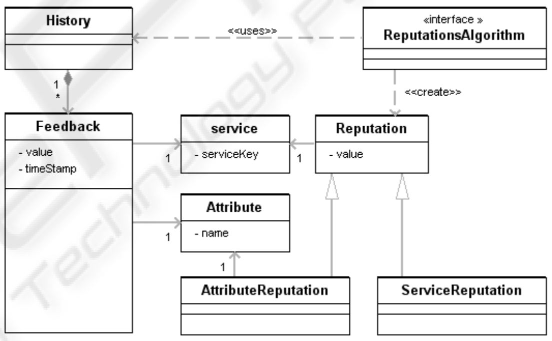

Figure 1: Conceptual feedback model.

The feedback model is based on the conceptual model of reputation of Maximilien et al. (Maximilien et al., 2002), which allows rating each service's attribute separately instead of the whole service. This feedback information is defined as a 4-tuple, consisting of the service and the service attributes which were rated, the evaluation value as well as the time stamp. All feedback information is stored in the history, which is used to calculate reputations based on reputation algorithms. A reputation can be related to a service or just to a service attribute. We call the first service reputation and the latter attribute reputation.

The feedback model of X-Fee is independent of the feedback category. Thus it can be used to save monitoring as well as user feedback information.

4 X-Fee – AN EXTENSIBLE FEEDBACK FRAMEWORK

X-Fee is a modular framework based on OSGi[1] and consists of five main parts: (1) a database to store all feedback information, (2) the monitoring feedback component, called SloMon, (3) the user feedback component, called WebRat, (4) a web service interface (WS) to support RPC-based access and (5) a message-oriented-middleware (MoM) to support event-based access to X-Fee.

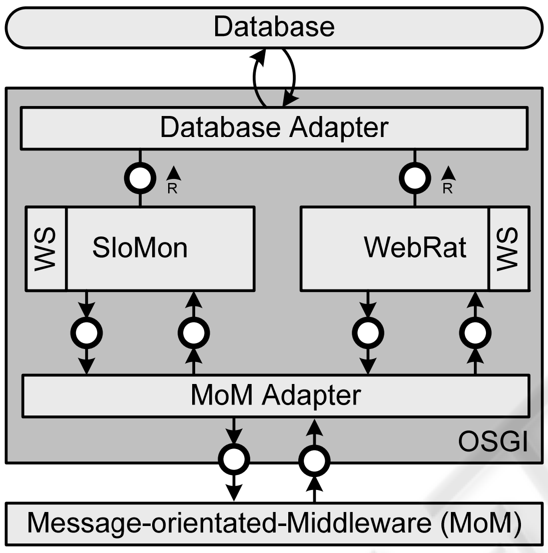

Figure 2: The overall architecture of X-Fee.

As it was already mentioned, one of the main disadvantage of state-of-the-art feedback components is the lack of flexibility and interoperability.

Interoperability is defined as the ability of a software module to run on and communicate with heterogeneous environments. X-Fee supports this design principle as follows:

1. The OSGi container keeps X-Fee independent from any target platform.
2. The web service interfaces of SloMon and WebRat allow a programming language independent RPC-based access to add and get feedback as well as reputation values.
3. The message-oriented-middleware allows SloMon and WebRat sending and receiving XML-based events in a programming language independent way.

Flexibility is defined as a system's ability to be adaptable to customers' needs, by e.g. changing or removing system's modules. X-Fee supports this feature as follows:

1. Both, the database as well as the message-oriented-middleware are integrated via adapters to make them replaceable.
2. Each part of X-Fee is implemented by two separate OSGi bundles. One for the interfaces and one for the implementation. Hence, the customer can adapt the system by exchanging the implementation bundle.
3. The modularity of OSGi allows using SloMon and WebRat separately as well as in an integrated way.

The vision of X-Fee is to provide a fully and ready-to use feedback framework. Thus we are realizing each part of X-Fee by a default implementation. Therefore MySQL[2] is used as database and the Apache ActiveMQ[3] as Message-oriented Middleware (MoM). These open-source third party components are not included in X-Fee and have to be installed separately.

4.1 Monitoring Feedback with SloMon

SloMon, depicted in Figure 3, provides a framework as well as a default implementation for SLO-monitoring. Its name stands for SLO monitoring. SloMon, which reuses the design pattern of Grand SLAM (Spillner et al, 2009), consists of three main parts: (1) the core, (2) the sensors and (3) the aggregators.

The *core* provides the web service interface and controls each part of SloMon. Therefore it is composed of three services: (1) the agreement service, (2) the sensor service and (3) the aggregator service.

The *agreement service* is responsible for reading agreements and for storing them into the database. The default implementation is based on the agreement standard Web Service Agreement (Andrieux et. al, 2007). Furthermore the agreement service allows loading feedback values as well as reputations from database via web service calls.

The *sensor service* manages the *sensors*, which are responsible for providing the feedback information. In terms of SLO monitoring feedback information are related to QoS measurement and SLO violation recognition, whereas each sensor takes care for one QoS property. Both measured

[1] http://www.osgi.org

[2] http://www.mysql.com/

[3] http://activemq.apache.org/

values as well as recognized SLO violations are published on the MoM to be available for interested components, such as error compensation strategies.

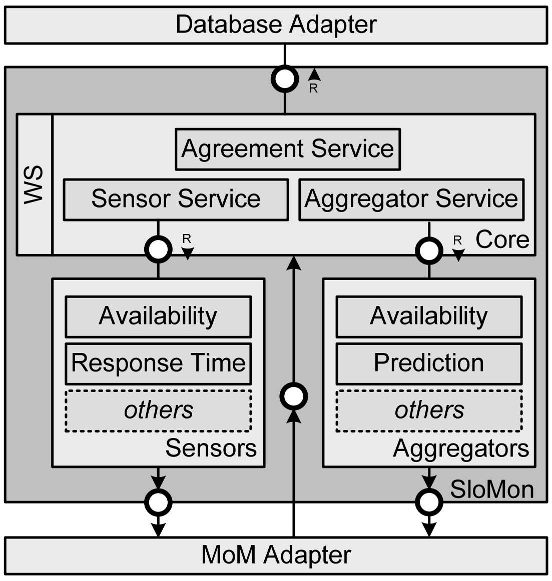

Figure 3: The architecture of SloMon.

SloMon supports two kinds of sensors: (1) periodical sensors and (2) invoke sensors. *A periodical sensor* measures the QoS property at regular intervals and returns the value to the sensor service, which stores the feedback information in the database. Therefore, at each interval, the sensor service reads out all agreements that are saved in the database and starts for each SLO a new instance of the registered periodical sensor, if such one exists.

An *invoke sensor* measures the QoS property during service invocation. Therefore the sensor service listens to the MoM to ServiceStarted events, which have to be generated by the infrastructure with which X-Fee is integrated. As soon as a ServiceStarted event occurs, the sensor service reads out all SLOs the started service guarantees and activates the appropriate sensor instance. The measurements are finished as soon as a ServiceStopped event for the running service is received on the MoM. The new feedback values returned by the sensors are saved in the database.

By default SloMon supports two types of sensors: (1) a response time sensor and (2) an availability sensor.

The *aggregator service* is responsible for the *aggregators*, which implement reputation algorithms, such as a service's availability per specific time period, e.g. per day. Similar to the sensors, each aggregator takes care of one reputation value. SloMon distinguishes between two kinds of aggregators: (1) service aggregators, providing service reputations and (2) attribute aggregators creating reputations for a specific attribute, i.e., QoS property.

All Aggregators work in parallel and are triggered by the aggregator service as soon as a new feedback value was inserted in the database. Therefore, the aggregator service reads out all registered services as well as their related QoS properties and starts a new instance of the appropriate aggregator. The new instance gets the list of all measurements related to the service or service attribute, calculates the reputation value and returns it to the aggregator service in order to be stored in the database.

The default implementation of SloMon contains two types of aggregators: (1) an availability aggregator providing the service's availability per hour, day, week, month as well as year and (2) a SLO violation prediction aggregator, which forecasts the probability with which a service will violates its SLO for response time and availability.

In contrast to the state-of-the-art monitoring components, SloMon is built to be highly flexible and extensible. Flexibility and extensibility allow changing, removing and adding modules to adapt the system to customer's needs. SloMon supports these features by implementing the services of the core component as well as each sensor and aggregator as separate bundles. This allows:

1. Changing the default implementation by removing the old bundle and adding a new one.
2. Adding a new sensors and aggregator, by adding a new bundle implementing the appropriate interface and registering the sensor or aggregator at SloMon.
3. Removing unnecessary sensors and aggregators by removing the respective bundles.

Due to OSGi's Hot Deployment, the adaptation of SloMon is possible even if the system is running.

4.2 User Feedback with WebRat

WebRat, depicted in Figure 4, provides a framework as well as a default implementation to deal with user feedback in the IoS. Its name stands for web service rating. WebRat consists of two main parts: (1) the core and (2) the aggregators.

The *core* controls each part of WebRat and is composed of two services: (1) the rating service and (2) the aggregator service.

The *rating service* implements a generic web service interface to add new feedback information, which means in terms of user feedback, to submit user ratings for one or more service attributes. The user ratings are stored in the database and distributed in the MoM as XML messages in order to inform components that are interested in. Furthermore it is possible to access the reputations, related to services and services attributes as well as to register services and their rateable attributes.

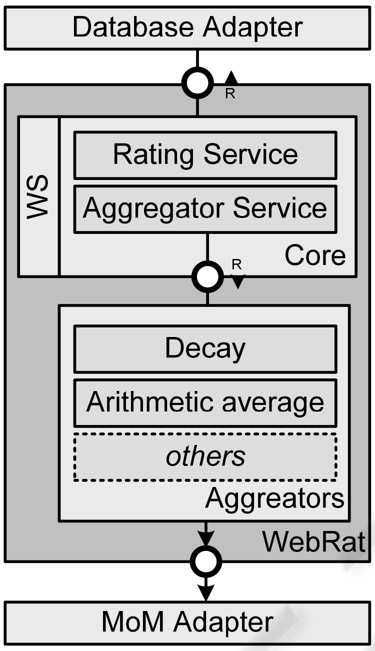

Figure 4: Architecture of WebRat.

The *aggregation service* manages, similar to SloMon, the service- and attribute-aggregators, which calculate the reputations based on appropriated algorithms. All aggregators work in parallel and are started as soon as a new user rating was inserted in the database. Hence, the aggregator service reads out all stored services and their attributes. For each service a new instance of each registered service aggregator is started. For each service attribute a new instance of the appropriate attribute aggregator is activated, as long as such an aggregator is registered at WebRat. Each aggregator instance gets the list of user ratings stored for the service or service attribute it manages. Based on this list, the reputation is deduced and returned to the aggregator service, which distributes the new value in the MoM and inserts it into the database.

By default WebRat supports two kind of service aggregators: (1) a simple arithmetic average and (2) a reputation algorithm based on the decay effect (Xu et al., 2007).

Analogous to SloMon, WebRat is highly flexible and extensible, by implementing the core component's services as well as each aggregator as separate bundles. This allows:

1. Changing the default implementation by removing the old bundle and adding a new one.
2. Adding new aggregator bundle which simply contains the algorithm to calculate the reputation based on a list of user ratings. All the other necessary tasks, such as database or MoM access, are done by the framework.
3. Removing unnecessary aggregators by removing the respective bundles.

5 DISCUSSION

The vision of X-Fee, which is still under development, is to be an interoperable, flexible and extensible infrastructure for providing feedback in the IoS. We fulfil theses requirements as follows:

1. Choosing OSGi as basis
2. Installing web service interfaces and MoM
3. Implementing all parts of X-Fee as modules, which are as small as possible and can be added, removed and changed corresponding to the user's needs

Despite the advantages discussed above, X-Fee has also two main weaknesses: (1) the web service interface and (2) the default implementation.

The web service interface constrains the performance because of the overhead caused by the XML marshalling and un-marshalling and the time needed to establish and close HTTP connections for every web service request. We are currently looking for alternative ways to provide interoperable RPC-based interfaces.

The second weakness, the default implement-tation, provides only simple algorithms, which do not conform to the current state-of-the-art and missing important aspects, including security. Thus the usefulness of X-Fee is limited without addressing these issues. But the missing parts can be implemented easily be replacing or adding new bundles. We plan to provide X-Fee as an open-source framework to be freely available.

6 CONCLUSIONS

This paper introduced X-Fee, an OSGi-based framework to provide monitoring and user feedback in the Internet of Services. X-Fee owns two main components: (1) SloMon and (2) WebRat, which can be used separately or in an integrated way to implement techniques for feedback creation as well as reputation calculation. In contrast to the state-of-the-art feedback components, X-Fee is interoperable, highly flexible and extensible and can be used by heterogeneous applications. Moreover, it is adaptive to the customers need.

ACKNOWLEDGEMENTS

The presented work is part of the research project THESEUS[4] and will be evaluated in the use case TEXO. 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., Franch, X., 2008. Service Level Agreement Monitor (SALMon). In Proceedings of the 7th *IEEE International Conference on Composition-Based Software Systems (ICCBSS),* Madrid, Spain, February 25-29, IEEE Computer Society, pp 224-227.

Andrieux, A., Czajkowski, K., Keahey, A., Ludwig, H., Nakata T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., 2007. Web Service Agreement Specification (WS-Agreement). *GFD-R-P.107*, Open Grid Forum, http://www.ogf.org/documents/GFD.107.pdf

Fickas, S., Feather, M. S., 1995. Requirement Monitoring in Dynamic Environments. In Proceedings of the 2nd *International Symposium on Requirements Engineering*, York, England, March 27-29. IEEE Computer Society, pp. 140.

Kalepu, S., Krishnaswamy, S, Loke, S. W., 2004. Reputation = f(User Ranking, Compliance, Verity). In Proceedings of the *IEEE International Conference on Web Services (ICWS'04)*, San Diego California, USA, July 6-9. IEEE Computer Society, pp. 200- 207.

Maximilen, E. M., Singh, P. M., 2002. Conceptual Model of Web Service Reputation. *ACM SIGMOD Record, SPECIAL ISSUE: Special section on semantic web and data management*. ACM, pp. 36-41.

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

Raimondi, F., Skene, J., Emmerich, W.:, 2008. Efficient Online Monitoring of Web-Service SLAs, In Proceedings of the *16th ACM SIGSOFT International Symposium on Foundations of software engineering,* Atlanta, Georgia, November 9-14, CAM, pp. 170-180.

Robinson, W. N., 2003. Monitoring Web Service Requirements, In Proceedings of the *11th IEEE International Conference on Requirements Engineering*, Monterey Bay, California, USA, September 8-12, IEEE Computer Society, pp. 65.

Spillner, J., Hoyer, J., 2009. SLA-Driven Service Marketplace Monitoring with Grand Slam. In Proceedings of the *4th International Conference on Software and Data Technologies (ICSOFT09)*, Sofia, Bulgaria, July 26-29.

Strunk A., Braun I., Reichert, S., Schill A., 2009. Supporting Rebinding in BPEL processes. In Proceedings of the *IEEE International Conference of Web Services (ICWS'09)*. Los Angeles, California, USA, July 6-10, IEEE Computer Society, pp. 864-871.

Stathel, S., van Dinther, C., Schönefeld, A. 2009. Service Innovation with Information Market. In Proceedings of the *9th International Conference on Business Informatics (Business Services: Concepts, Technologies, Applications)*, February 25-27, Vienna, Austria, pp. 825-834.

Wang, Y., Vassileva, J., 2007. Towards trust and reputation based web service selection: A survey. In Proceedings of *the 27th International Conference in Distributed Computing Systems Workshop (ICDCSW)*, Toronto, Canada, June 22-29, pp. 25.

Xu, Z., Martin, P., Powley, W., Zulkernine, F., 2007. Reputation-Enhanced QoS-based Web Service Discovery. In Proceedings of the *IEEE International Conference on Web Services (ICWS'07),* Salt Lake City, Utah, USA, July 9-13, IEEE Computer Society, pp. 249-256.

[4] http://www.theseus-programm.de