Kingdom), CWI (Netherlands), University of L'Aquila (Italy), Technalia (Spain), Softeam (France), and Uninova and Unparallel Innovation (Portugal). OSSMETER aims to: extend the state-of-the-art in the field of automated analysis and the measurement of OSS; and develop a platform that will support decision makers in the process of discovering, comparing, assessing and monitoring the health, quality, impact and activity of OSS. To achieve these goals OSSMETER will develop trustworthy quality indicators by providing a set of analysis and measurement components. More specifically, the OSSMETER platform will provide the following software measurement components [2]:

- programming language-agnostic and language-specific components to assess a project's source code quality;
- text mining components to analyse the natural language information extracted from communication channels and bug-tracking systems and thus, provide information about communication quality within the project and community activity (around the project); and
- software forge mining components to extract project-related metadata.

After this data is extracted by the platform, it will be made available via a public web application, along with its decision support tools and visualizations.

With almost six months to go, the OSSMETER team is planning to release a beta-version of the platform to the project's industrial partners to support the analysis of OSS projects hosted on SourceForge, GitHub and Eclipse forges. The OSSMETER platform is scheduled to be publicly available at the beginning of 2015. The OSSMETER team will run a public version of the platform which will analyse a number of OSS projects and the results will be published on the OSSMETER website. Furthermore, the platform will also be available for download for personal analyses.

Link:
OSSMETER project:
http://www.ossmeter.org/

Reference:
[1] N. Matragkas et al.: "OSSMETER D5.1 - Platform Architecture Specification", Technical Report, pp. 1-32, Department of Computer Science, University of York, York, UK, 2013 (http://www.ossmeter.eu/publications)

Please contact:
Nicholas Matragkas
University of York
Tel: +44 (0)1904 325164
E-mail: nicholas.matragkas@york.ac.uk

Monitoring Services Quality in the Cloud

by Miguel Zuñiga-Prieto, Priscila Cedillo, Javier Gonzalez-Huerta, Emilio Insfran and Silvia Abrahão

Due to the dynamic nature of cloud computing environments, continuous monitoring of the quality of cloud services is needed in order to satisfy business goals and enforce service-level agreements (SLAs). Current approaches for SLA specifications in IT services are not sufficient since SLAs are usually based on templates that are expressed in a natural language, making automated compliance verification and assurance tasks difficult. In such a context, the use of models at runtime becomes particularly relevant: such models can help retrieve data from the running system to verify SLA compliance and if the desired quality levels are not achieved, drive the dynamic reconfiguration of the cloud services architecture.

Cloud computing represents much more than an infrastructure with which organizations can quickly and efficiently provision and manage their computing capabilities. It also represents a fundamental shift in how cloud applications need to be built, run and monitored. While some vendors are offering different technologies, a mature set of development tools that can facilitate cross-cloud development, deployment and evaluation is yet to be developed. This definitely represents a growth area in the future. The different nature of cloud application development will drive changes in software development process frameworks, which will become more self-maintained and practice-oriented.

Cloud services need to comply with a set of contract clauses and quality requirements, specified by an SLA. To support the fulfillment of this agreement a monitoring process can be defined which allows service providers to determine the actual quality of cloud services. Traditional monitoring technologies are restricted to static and homogenous environments and, as such, cannot be appropriately applied to cloud environments [3]. Further, during the development of these technologies, many assumptions are realized at design time. However, due to the dynamic nature of cloud computing, meeting those assumptions in this context is not possible. It is necessary, therefore, to monitor the continuous satisfaction of the functional and quality requirements at runtime.

During this monitoring process, the violation of an SLA clause may trigger the dynamic reconfiguration of the existing cloud services architecture. Dynamic reconfiguration creates and destroys architectural elements instances at runtime: this is particularly important in the context of cloud computing as their services must continue working while the reconfiguration takes place. However, little attention has been paid to supporting this reconfiguration at runtime and only recently has the field of software engineering research started focusing on these issues [1].

Through the Value@Cloud project, funded by the Ministry of Economy and Competitiveness in Spain, we are developing a framework to support model-driven incremental cloud service development. Specifically, the framework supports cloud development teams to: i) capture business goals and Quality-of-Service (QoS) attributes (which will form part of the SLA); ii) create and incrementally deploy architecturual-

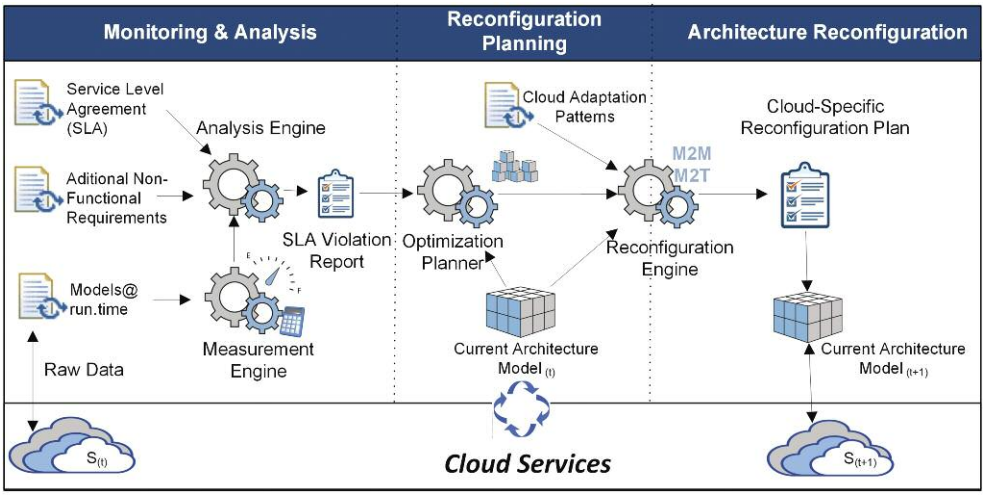

Figure 1: Cloud services quality monitoring and reconfiguration infrastructure.

centric cloud services that are capable of dynamically evolving; and iii) monitor the quality of cloud services delivered to the customers.

The monitoring strategy developed through this project is based on two key elements. The first is models at runtime [2] which verify the degree of compliance against the quality requirements specified in the SLA. The second is techniques for dynamically reconfiguring the cloud services architecture if the desired quality levels are not satisfied. The main activities and artifacts involved in this monitoring strategy are shown in Figure 1.

Models at runtime offer flexibility to the monitoring infrastructure through their reflection mechanisms: the modification of quality requirements may dynamically change the monitoring computation, thus avoiding the need to adjust the monitoring infrastructure. In our approach, models at runtime are part of a monitoring & analysis middleware that interacts with cloud services. This middleware retrieves data in the model at runtime, analyzes the information and provides a report outlining the SLA violations. This report is used in the reconfiguration planning to dynamically reconfigure the cloud services architecture in order to satisfy the SLA. The architecture reconfiguration is carried out by generating cloud specific reconfiguration plans, which include adaptation patterns to be applied to cloud service instances at runtime.

We believe that our approach will facilitate the monitoring of the higher-level quality attributes specified in SLAs. It can also provide the architect with flexibility if new quality requirements need to be added or modified since the changes will be performed at runtime and the monitoring infrastructure will remain unchanged. Finally, not only does this approach report the SLA violations identified but also provides a reconfiguration plan for dynamically changing the cloud service architecture in order to satisfy the SLA quality requirements.

Link:
ISSI Research Group at Universitat Politècnica de València:
http://issi.dsic.upv.es/projects

References:
[1] L. Baresi, C. Ghezzi: "The Disappearing Boundary Between Development-time and Run-time", FSE/SDP Workshop on Future of software Engineering Research, pp. 17-21, 2010
[2] N. Bencomo et al.: "Requirements reflection: requirements as runtime entities", 32nd International Conference on Software Engineering, pp. 199-202, 2010
[3] V.C. Emeakaroha et al.: "Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments", HPCS 2010, pp. 48-54.

Please contact:
Silvia Abrahão
Universitat Politècnica de València, Spain
Email: sabrahao@dsic.upv.es

Dictō: Keeping Software Architecture Under Control

by Andrea Caracciolo, Mircea Filip Lungu and Oscar Nierstrasz

Dictō is a declarative language for specifying architectural rules that uses a single uniform notation. Once defined, the rules can automatically be validated using adapted off-the-shelf tools.

Quality requirements (e.g., performance or modifiability) and other derived constraints (e.g., naming conventions or module dependencies) are often described in software architecture documents in the form of rules. For example:

- "Repository interfaces can only declare methods named 'find*()'"; or
- "If an exception is wrapped into another one, the wrapped exception must be referenced as the cause"; and
- "Entity bean attributes of type 'Code' must be annotated with @Type(type = "com.[..].hibernate.CodeMapping")".

Ideally, rules such as these should be checked periodically and automatically. However, after interviewing and sur-