# Framework for Dynamic Architecture Reconfiguration of Cloud Services

Miguel Zuñiga-Prieto, Javier Gonzalez-Huerta, Silvia Abrahao, Emilio Insfran

ISSI Research Group, Department of Information Systems and Computation
Universität Politècnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
{mzuniga, jagonzalez, sabrahao, einsfran}@dsic.upv.es

## 1 Motivation and Goals

Cloud computing is a paradigm that is transforming the way in which organization acquire computational resources and is receiving more attention from the research community. The incremental deployment of cloud services is particularly important in agile development of cloud services, where successive cloud service increments must be integrated into existing cloud service architectures. This requires dynamic reconfiguration of software architectures, especially in cloud environments where services cannot be stopped in order to apply reconfiguration changes. A model-driven architecture reconfiguration process uses models to represent architectural and technological concepts at a high level of abstraction. Then transformations are applied on models to add the necessary detail in order to generate specific reconfiguration operations. This demands development efforts that could be alleviated not only by defining the models and transformation sequences, but also by providing tools that facilitate tasks.

The aim of this research work is to propose a framework to facilitate the dynamic architectures reconfiguration of cloud services, triggered by the incremental deployment of cloud services. The aforementioned aim will be satisfied by dealing with the following sub-goals, which are:

- Provide a set of tools to support the process of dynamic software architecture reconfiguration triggered by the deployment of new cloud services.
- Define transformation chains that allow to obtain platform specific reconfiguration plans, starting from a high level description of how the architecture of a software increment affect the current cloud service architecture.

In the next section, we introduce important adaptation concepts applied in this work.

## 2 Background information

In this section, due to space limitations, we just include the description of important Software Adaptation Techniques applied in this framework.

Software adaptation patterns represent generic and repeatable solutions to manage change in recurring architectural adaptation problems, and prescribe the steps needed

to dynamically adapt a software system at runtime from one configuration to another [1]. The use of adaptation patterns is a trend to support reuse in evolution for dynamic adaptive software architecture [2]. Adaptation of software architectures is not only supported by change management proposals, but also by proposals for solving the problems that arise when the interacting entities do not match properly. Software adaptation promotes the generation of software adaptors to bridge incompatibilities among services (e.g., different names of methods and services, different message ordering, etc.) in an nonintrusive way [3,4, 5]. Generation techniques for software adaptors are beginning to be used in cloud environments [6].

Cloud applications integrate and compose different cloud services. The cloud services to be integrated may come from the delivering of a software increment in an incremental development approach, or just may be product of maintenance/evolution phases. When we refer to a software increment, we mean one or more cloud services than are included in a software increment that need to be effectively deployed.

## 3 Proposed Approach and Contributions

This poster presents a model-driven framework that provides models and tools that facilitate the dynamic architecture reconfiguration activities followed during the integration of cloud services. Models allow representing high-level description of how the *Architecture of a Software Increment (ASI)* affects the current cloud service architecture. Transformation chains, establish how to apply consecutive transformations in order to generate platform specific reconfiguration plans, obtained starting from models.. Finally, tools not only alleviate tasks of specification of models and design of transformations, but also provide dedicated services to be used during reconfiguration.

### 3.1 Reconfiguration Process

The process to which the proposed framework will give support aims to help software developers during the deployment phase, on activities related to integration of software increments into existing services in the cloud. This process supports the integration from an architectural point of view. Its activities support the management of dynamic reconfiguration of existing cloud services architectures, produced due to the integration of architectural elements corresponding to the *ASI*. The main activities of the process are (see Fig. 1): i) *Specify Increments*; ii) *Check Increment Compatibility*; and iii) *Reconfigure Architecture*. Below we present a process overview.

In the first activity, *Software Architects* specify the *ASI* as well as the impact of the integration on the current cloud service architecture. The *ASI* includes: i) information about structure and behavior of cloud services included in the software increment; ii) information about how elements of the ASI collaborate to reconfigure the current architecture; and iii) information about related aspects of architectures of cloud services [7]. Next, in the following activity, *Software Architects* participate in verifying if the *ASI* is compatible with the current cloud service architecture. If discrepancies exist be-

tween interfaces of these architectures and it is possible to solve them, *Software Architects* apply model-to-text (M2T) transformations to generate skeletons of *Software Adaptors* specific for a cloud platform technology. *Software Developers* complete *Software Adaptors* skeletons implementing code to solve discrepancies. Finally, in the last activity, *Software Architects* participate in selecting the adaptation patterns best suited to integrate *ASI* elements into the current cloud service architecture. Once the adaptation patterns have been selected, *Software Architects* apply model transformations to generate *Reconfiguration Plan Specific of Cloud Provider*that operationalize those adaptation patterns according to the *ASI*. In the last step of this activity *Cloud Specialist*, expert in deployment, integrates the *ASI* into the current architecture by i) deploying the *Software Adaptors*, and ii) using dedicated services to apply the *Reconfiguration Plan Specific of Cloud Provider* in corresponding cloud platform. The integration, dynamically reconfigures instances of the running *Actual System Architecture*.

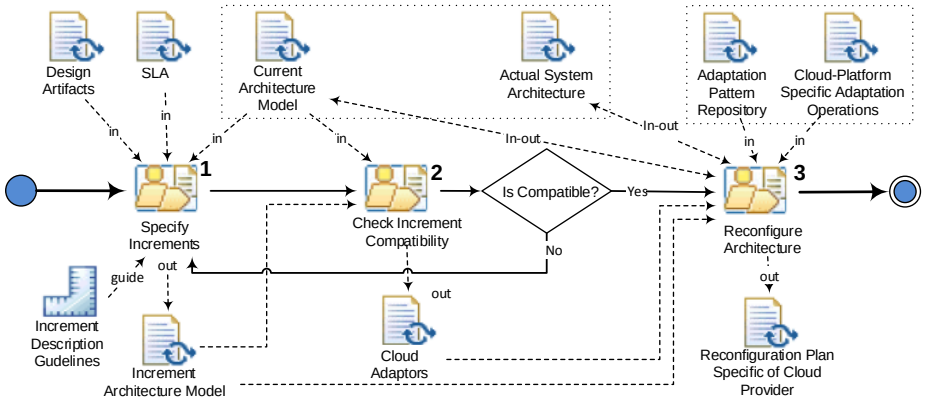


**Fig. 1.** Overview of the reconfiguration process

### 3.2 Increment Description Language

To support the first activity of the dynamic architecture reconfiguration process (see Section 3.1), this framework provides to Software Architects a specialized language to specify the architecture of the software increment. We call it *Increment Description Language (IDL)*. Service Oriented Architecture Modeling Language (SoaML)[8] leverages Model Driven Architecture (MDA) and provides a UML profile and meta-model for the specification of services. However, SoaML does not allow to represent how the architecture of a software increment affects the existing cloud architecture nor to specify information related to cloud software architectures. *IDL* is a Domain-Specific Language (DSL) that extends SoaML, so *IDL* meta-model extends the SoaML meta-model using stereotypes to point out the way in which each element of ASI impacts on current cloud service architecture. Furthermore, we use tagged values to specify information related to cloud software architectures.

### 3.3 Transformation Chains

To support the second and third activity of the dynamic architecture reconfiguration process (see Section 3.1), this framework identify a set of models and transformation chains (consecutive transformations) that must be applied to those models. This, in order to generate platform specific reconfiguration plans, obtained starting from a high-level description of how the architecture of a software increment affects the current cloud service architecture.

## 4 Contributions

A framework for dynamic architecture reconfiguration that properly combines Model-Driven and Software Adaptation Techniques represents an important step towards the incremental and dynamic deployment of cloud services into existing cloud service architectures.

The first contribution is an *Increment Description Language* that allows *Software Architects* to specify the architecture of the software increment. They will be able to specify architecture reconfiguration operations (e.g., add service, add connector, etc.) using a high-level abstraction language. To put it another way, when a *Software Architect* specifies the impact of ASI on current architecture, what he/she is really doing is specifying architecture reconfiguration operations at a high-level of abstraction.

The second contribution is the definition of Transformation Chains, used to: i) promote compatibility between the architecture of the increment with the existing cloud architecture; ii) generate a *Reconfiguration Plan Specific of Cloud Provider* that applies adaptation patterns to reconfigure existing cloud architecture.

Finally, in order to achieve reconfiguration we provide dedicated services. These services use the information of reconfiguration plans to dynamically reconfigure instances of the running Actual System Architecture.

## 5 Related work

Software evolution based on reconfiguration of software architectures is an active area of research; however, there are gaps that still need to be covered. Jamshidi et al. [2], identified a relative lack of contributions for runtime evolution as well as frameworks to support the reconfiguration process. They found lack of support during the integration and provisioning stage as well as during deployment stage. In our work, we give support to the dynamic reconfiguration of software architectures in the deployment stage of the software life cycle.

In this section, due to space limitations, we detail the gaps we found in most relevant related works [9, 10, 11]. Baresi et al.[9], MOdel-based SElf-adaptation of SOA systems (MOSES) [10], and Self-architecting Software Systems (SASSY) [11]. These works i) take into account structural and behavioral aspects for reconfiguration; ii) use SLA or QoS negotiation to discover and select the more suitable service implementation (instance); and iii) apply dynamic binding for reconfiguration. This means that

reconfiguration improves non-functional properties through perfective changes. However, adaptive changes (e.g., software increments due to new functionalities) that may require architecture reconfiguration are not taken into account. They also abstract models of business requirements or derive high level architectures; however, they do not take into account the importance of architectural aspects in incremental development processes [12]. Despite the fact that the cited approaches propose consistency or compatibility checking tasks, they do not provide solutions to support deployment on changing cloud platforms. In addition, those who propose frameworks or support architecture specification do not consider the architecture of the increment as an independent entity.

**Acknowledgments.** This research was supported by the Value@Cloud project (MICINN TIN2013-46300-R); the Scholarship Program Senescyt, Ecuador; the Faculty of Engineering, University of Cuenca, Ecuador; and the ValI+D program (ACIF/2011/235), Generalitat Valenciana.

## 6 References

1. Gomaa, H., Hashimoto, K., Kim, M., Malek, S., Menascé, D.: Software Adaptation Patterns for Service-Oriented Architectures. ACM Symposium on Applied Computing. pp. 462–469. ACM, New York (2010).
2. Jamshidi, P., Ghafari, M., Ahmad, A., Pahl, C.: A Framework for Classifying and Comparing Architecture-Centric Software Evolution Research. 17th European Conference on Software Maintenance and Reengineering. pp. 305–314. IEEE, Genova (2013).
3. Canal, C., Poizat, P., Salaun, G.: Model-Based Adaptation of Behavioral Mismatching Components. Softw. Eng. IEEE Trans. 34, 546–563 (2008).
4. Yellin, D.M., Strom, R.E.: Protocol Specifications and Component Adaptors. ACM Trans. Program. Lang. Syst. 19, 292–333 (1997).
5. Becker, S., Brogi, A., Gorton, I., Overhage, S., Romanovsky, A., Tivoli, M.: Towards an Engineering Approach to Component Adaptation. Springer Berlin Heidelberg (2006).
6. Miranda, J., Guillen, J., Murillo, J.M., Canal, C.: Assisting Cloud Service Migration Using Software Adaptation Techniques. 6th Int. Conf. on Cloud Computing. pp. 573–580 (2013).
7. Hamdaqa, M., Livogiannis, T., Tahvildari, L.: A Reference Model for Developing Cloud Applications. CLOSER. pp. 98–103. Citeseer (2011).
8. Object Management Group: Service Oriented Architecture Modeling Language (SoaML), http://www.omg.org/spec/SoaML/.
9. Baresi, L., Heckel, R., Thöne, S., Varr´o, D´.: Style-Based Modeling and Refinement of Service-Oriented Architectures. Softw. Syst. Model. 5, 187–207 (2006).
10. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: QoS-Driven Runtime Adaptation of Service Oriented Architectures. Proc. 7th Jt. Meet. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. 131–140 (2009).
11. Menascé, D.A., Gomaa, H., Malek, S., Sousa, J.P.: SASSY: A Framework for Self-Architecting Service-Oriented Systems. Software, IEEE. 28, 78–85 (2011).
12. Babar, M.A., Brown, A.W., Mistrik, I.: Making Software Architecture and Agile Approaches Work Together: Foundations and Approaches. Agile Software Architecture: ligning Agile Processes and Software Architectures. pp. 1–22. Morgan Kaufmann (2013).