Computers and Electrical Engineering 000 (2016) 1–12

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Pattern-based orchestration and automatic verification of composite cloud services

Flora Amato[a,*], Francesco Moscato[b]

[a] DIETI, University of Naples Federico II, Italy
[b] DiSciPol, Second University of Naples, Italy

ARTICLE INFO

Article history:
Received 29 August 2015
Revised 7 August 2016
Accepted 8 August 2016
Available online xxx

Keywords:
Cloud patterns
Verification
Semantics
Orchestration
Service level agreeement
Quality of service

ABSTRACT

Recent years have seen an increase of complexity in paradigms and languages for development of Cloud Systems. The need to build value added services and resources promoted pattern-based composition and orchestration as new hot research topics. Anyway, unlike web services, it is unclear what orchestration means for Cloud Systems. In this scenario, a way to automatically build composite services from their pattern-based description is appealing. In this work we describe a methodology for automatic composition and verification of Cloud Services which is driven by formal orchestration language.

1. Introduction

Cloud Architecture is nowadays a *de facto* standard for providing *any* kind of service on Internet. Big vendors, like Amazon Web Services (AWS)[1] or Microsoft with Azure[2] are definitely the main actors in Cloud Architecture definition. In [1] NIST extends the meaning of orchestration to Cloud Architecture. In addition, a new trend in Cloud Services design and management grew up in the last years: the definition of design patterns for Cloud Computing. Anyway, the introduction of design patterns in Cloud Computing requires a concept of orchestration that is more complex than the one defined in [1]: as we will show in this work many design patterns [2] with different purposes can be described as complex workflows. Workflow based composition opens the problem of understanding if a composite service is compliant with users' requirements and QoS. Properties of composite services obviously depend on components properties and composition. We think that compliance cannot be evaluated only by means of syntactical or type checking. Actually, several semantics-based approaches for *simple* web services composition exist (a survey is in [3]). Some of them exploit BPEL4WS [4] orchestration language and OWL-based ontologies for services description. In this context, it is clear that a methodology able to compose and verify Cloud Services by using Cloud Design Patterns is really appealing. This is the reason for we propose a formal orchestration language able to describe all elements, events and composition issues we need to describe complex services. The language enables pattern-based composition of Cloud elements at different layers. In this way, we reach two goals:

* Corresponding author.
E-mail addresses: flora.amato@unina.it (F. Amato), francesco.moscato@unina2.it (F. Moscato).

[1] https://aws.amazon.com
[2] https://azure.microsoft.com/

http://dx.doi.org/10.1016/j.compeleceng.2016.08.006
0045-7906/

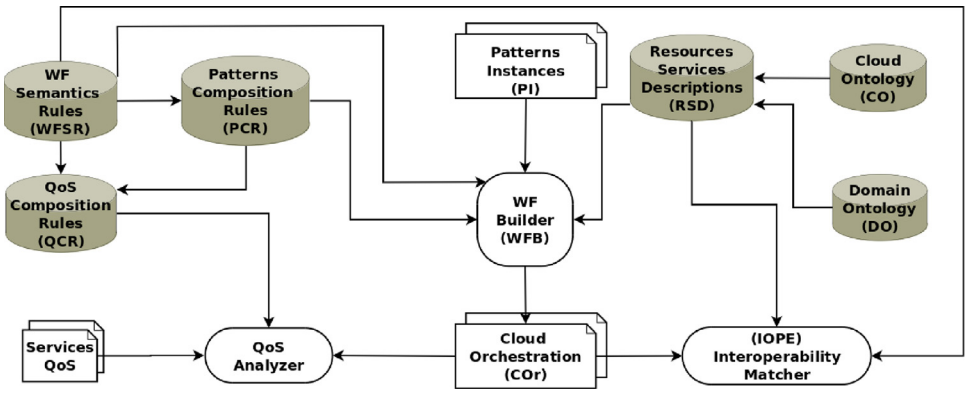

Fig. 1. System architecture.

first, our methodology enables Cloud designer and programmers to describe interactions of components directly by means of patterns; second, the proposed approach allows for the analysis of semantics and Quality of Services(QoS) of composite services and resources.

The paper is organized as follows. Section 2 contains the description of the methodology we propose and the architecture of a framework used to enact its steps. In order to describe patterns and composite services in Section 3 we introduce an orchestration language. Section 4 discusses the problem of binding real services into composite services. Section 5 proposes a full example and Section 6 contains an analysis of related works. Finally, Section 7 reports some concluding remarks.

2. Methodology and architecture

In the methodology we are going to illustrate, a formal workflow language describes composition and patterns. Its formal semantics, together with a semantic-based definition of cloud resources and services, enables automatic composition. Patterns give information about *how* services and resources interact at early design and development stages. This abstract information suggests the way to analyse, for example, composition soundness or quality of services. Let us consider a simple example with two basic control flow patterns: the *sequence* and the *1 out of N*. From an availability point of view, if N services are organized in a sequence, the failure probability of the composite service is the sum of the failure probabilities of components, on the other hand in the 1 out of N composition, it is their product. Our workflow language (Operational Flow Language: OFL in the following) has simple constructs that can be used to define complex Cloud Patterns[3] [2]. OFL is expressive enough to describe several patterns, as well as simple enough to be defined by a clear operational semantics. The final step is the management of Cloud composite services as patterns instances.

In order to bind real services in composition skeleton, we have to know their functionalities and their QoS properties. Our methodology supports ontology-based description of services by using OWL-S [5] but it is not enough expressive to characterize general composition [4]. We think that an Inputs, Outputs, Preconditions, Effects (IOPE [6]) characterization of Cloud resources, as well as a semantic description of what resources are and what they do, are useful to choose components that best match semantics and requirements of composite services. For proper matching and ontology [7] describes roles of available resources and services. Finally, for what QoS analysis and composition concerns, the workflow graphs described by OFL allow for the creation of analysis models by using Model Transformation techniques [8].

Fig. 1 depicts the architecture that enforces the methodology previously introduced.

The **WF Builder**, the **IOPE Interoperability Matcher** and the **QoS Builder** respectively manage goals of: (a) creating a workflow of composite services from their patter-based description; (b) matching component services in the orchestrated service; (c) analysing QoS of resulting composite Cloud Services. The WF (Workflow) Builder is based on a Knowledge Base of logic rules containing (a) the Operational semantics of OFL (in the repository **WF Semantic Rules**); (b) the operational semantics rules of orchestration patterns (**Patterns Composition Rules**). They are in turn defined by OFL. Results from WF Builder are skeletons used to implement Orchestrated Cloud Services. Proper back-end units can read skeletons in order to create stubs for different Vendors languages and APIs. Inputs for the Service Interoperability Matching are **Cloud Orchestration** skeletons (declaring how services interact) and the descriptions of component services and resources. Descriptions use OWL-S and IOPE grounding for cloud resources. In addition, the description of services functionalities depends on domain ontologies (i.e. financial services, E-Health etc.). Furthermore Cloud Ontology [7] describes the roles of elements in the Cloud Architecture. The description of these ontologies is out of the scope of this work. Finally, QoS Analyzer takes Cloud Orchestration and QoS descriptions in order to build analysable models (by using **QoS Composition rules**). At the moment the analyzer supports only performance and availability analyses.

[3] https://cloudpatterns.org, http://en.clouddesignpattern.org/index.php/Main_Page, https://msdn.microsoft.com/en-us/library/dn568099.aspx

Please cite this article as: F. Amato, F. Moscato, Pattern-based orchestration and automatic verification of composite cloud services, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.006