Computers and Electrical Engineering 000 (2017) 1–21

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Achieving autonomic Web service compositions with models at runtime☆

Germán H. Alférez[a,*], Vicente Pelechano[b]

[a] Facultad de Ingeniería y Tecnología, Universidad de Montemorelos, Apartado 16-5, Montemorelos N.L., Mexico
[b] Centro de Investigación en Métodos de Producción de Software (ProS), Universitat Politècnica de València, Camí de Vera s/n, E-46022, Spain

ARTICLE INFO

Article history:
Received 24 August 2016
Revised 7 August 2017
Accepted 7 August 2017
Available online xxx

Keywords:
Web service compositions
Models at runtime
Autonomic computing
Dynamic software product lines
Dynamic adaptation
Dynamic evolution

ABSTRACT

Several exceptional situations may arise in the complex, heterogeneous, and changing contexts where Web service operations run. For instance, a Web service operation may have greatly increased its execution time or may have become unavailable. The contribution of this article is to provide a tool-supported framework to guide autonomic adjustments of context-aware service compositions using models at runtime. During execution, when problematic events arise in the context, models are used by an autonomic architecture to guide changes of the service composition. Under the closed-world assumption, the possible context events are fully known at design time. Nevertheless, it is difficult to foresee all the possible situations arising in uncertain contexts where service compositions run. Therefore, the proposed framework also covers the dynamic evolution of service compositions to deal with unexpected events in the open world. An evaluation demonstrates that our framework is efficient during dynamic adjustments.

1. Introduction

In nature, adaptation contributes to the survival of individuals to cope with the weather, enemies, or any hazards. As organisms live in intricate, changing environments, software is executed in complex, heterogeneous, and highly-intertwined computing infrastructures in which a diversity of events may arise. For example, security threats, network problems, performance reduction in one of the servers, etc. Thus, it is desirable to translate the ideas of adaptation in the natural world to software in order to solve these situations.

In fact, adaptability is emerging as a necessary underlying capability of highly-dynamic context-aware systems. The *context* is any information that can be used to characterize the situation of an entity [1]. Context-aware systems are concerned with the acquisition of context, the abstraction and understanding of context, and application behavior based on the recognized context. Adaptability is specially relevant in systems built upon Web service operations. Several exceptional situations may arise in the complex, heterogeneous, and changing contexts where they run. For instance, a Web service operation may have greatly increased its execution time or may have become unavailable. Cases like these make evident the need for dynamic adaptations in critical systems that are based on Web service compositions (or simply called service composi-

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. S. Kallel.

* Corresponding author.
E-mail addresses: harveyalferez@um.edu.mx, harveyalferez@gmail.com (G.H. Alférez), pele@dsic.upv.es (V. Pelechano).

http://dx.doi.org/10.1016/j.compeleceng.2017.08.004
0045-7906/

tions). However, implementing dynamic adaptations with variability constructs at the language level can become complex and error-prone, specially in large systems.

This article provides a framework based on models at runtime to guide dynamic adjustments of context-aware service compositions. *Models at runtime* can be defined as causally connected self-representations of the associated system that emphasize the structure, behavior, or goals of the system from a problem space perspective [2]. Our framework tries to solve the following gaps found in related work on autonomic service compositions: 1) need for designing the autonomic behavior of context-aware service compositions by means of easy-to-understand abstractions; 2) need for facing unanticipated context events (unseen at design time) in the open world; 3) need for transparent solutions that do not require changes in the orchestration engine; and 4) need for means of verifying autonomic adjustments to achieve safe reconfigurations.

In our approach, in response to changes in the context, the system itself can query models during execution to determine the necessary modifications in the underlying service composition. When a problematic event arises in the context, models are leveraged for decision-making. The activation and deactivation of features in a variability model result in changes in a composition model that abstracts the service composition. Our approach verifies new possible configurations before they are applied to the running service composition. Changes in the models are reflected into the service composition by adding or removing fragments of Web Services Business Process Execution Language (WS-BPEL) code, which can be deployed at runtime. Our solution does not require the modification of the orchestration engine. If model adaptations are not enough to solve uncertainty at runtime, we provide a solution to guide the dynamic evolution of the supporting models to preserve high-level requirements. We present novel evaluation results of our framework that demonstrate its potential.

The rest of this article is organized as follows: Section 2 presents the state of the art. In this section, a taxonomy is proposed to facilitate the analysis of related work. Section 3 describes the main building blocks of our framework. Section 4 presents our framework for autonomic service compositions. Section 5 presents the evaluation of our framework. Section 6 presents conclusions and future work.

2. Research on autonomic service compositions

This section examines related work proposed during a period of thirteen years to achieve autonomic service compositions in the following areas: variability constructs at the language level, brokers, models at runtime, and dynamic software product lines (DSPLs). These areas are covered in this section because of their relevance in the state of the art of autonomic service compositions and their closed relationship with our approach. The aim of studying these approaches is to answer the following questions:

- `Question 1 (Q1)`: Which approaches have been proposed in recent years to achieve autonomic service compositions by means of variability constructs at the language level, brokers, models at runtime, and DSPLs?
- `Question 2 (Q2)`: How to characterize the approaches for autonomic service compositions according to desirable properties of Self-Adaptive Systems (SAS)?
- `Question 3 (Q3)`: What are the gaps to be faced in the field of autonomic service compositions?

In order to answer Q1, we identified **31 approaches** on autonomic service compositions classified in the aforementioned areas of research (see Table 1). WS-BPEL has become the de facto language to orchestrate service compositions. However, there are two main drawbacks when using WS-BPEL in enterprise systems: 1) WS-BPEL has an inherent static nature; and 2) WS-BPEL does not provide any means for monitoring the context. Therefore, research works in the first area are particularly focused on extending WS-BPEL to guide dynamic adjustments according to collected context data. Intrusive and non-intrusive strategies have been proposed to specify these extensions. On one hand, an intrusive strategy has been to define variation points, variants, and configurations for a process inside the service composition specification (i.e., the composition schema). On the other hand, non-intrusive strategies have been proposed by means of aspect orientation, Event-Condition-Action (ECA) rules, and language-based monitoring and recovery strategies.

In another area of research, a system works as a broker that is in charge of selecting or binding partner service operations at runtime. To this end, the broker intercepts service messages. Then, it exchanges service operations by invoking the most adequate set of service operations (e.g. according to Quality of Service (QoS) attributes) to accomplish a task.

In the third area of research, models at runtime extend the applicability of models produced in Model-driven Engineering (MDE) approaches to execution time. Since models at runtime provide up-to-date and exact information about the runtime system, they can offer a rich semantic base for runtime decision-making in order to achieve system adaptation. Thus, the system itself can query the models at runtime to make adaptation decisions, to choose the adaptation strategy, and to control the adaptation process.

Also, Dynamic Software Product Line Engineering (DSPLE) has been proposed as a way to achieve autonomic service compositions. In DSPLs, variation points are bound initially when the software is launched to adapt to the current context, as well as during operation to adapt to changes in the context [34].

According to Table 1, brokers have been the most widely used mechanism to achieve autonomic service compositions among the areas covered in this section. Also, it is interesting to see that variability constructs at the language level were a popular way to achieve autonomic service compositions during several years (from 2006 to 2011). Nevertheless, this interest has moved in recent years towards models at runtime and DSPLE.

Please cite this article as: G.H. Alférez, V. Pelechano, Achieving Autonomic Web Service Compositions with Models at Runtime, Computers and Electrical Engineering (2017), http://dx.doi.org/10.1016/j.compeleceng.2017.08.004