Journal of Computer and System Sciences 81 (2015) 533–552

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Model-driven development of adaptive web service processes with aspects and rules

Jian Yu [a,*], Quan Z. Sheng [b], Joshua K.Y. Swee [b], Jun Han [c], Chengfei Liu [c], Talal H. Noor [b]

[a] School of Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand
[b] School of Computer Science, The University of Adelaide, Adelaide, Australia
[c] Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 19 December 2013
Received in revised form 31 May 2014
Accepted 30 July 2014
Available online 20 November 2014

Keywords:
Web services
Adaptive systems
Model-driven development
Aspect-oriented methodology
Design tools and techniques

ABSTRACT

Modern software systems are frequently required to be adaptive in order to cope with constant changes. Unfortunately, service-oriented systems built with WS-BPEL are still too rigid. In this paper, we propose a novel model-driven approach to supporting the development of dynamically adaptive WS-BPEL based systems. We model the system functionality with two distinct but highly correlated parts: a *stable* part called the base model describing the *flow logic* aspect and a *volatile* part called the variable model describing the *decision logic* aspect. We develop an aspect-oriented method to weave the base model and the variable model together so that runtime changes can be applied to the variable model without affecting the base model. A model-driven platform has been implemented to support the development of adaptive WS-BPEL processes. In-lab experiments show that our approach has low performance overhead. A real-life case study also validates the applicability of our approach.

1. Introduction

Service-Oriented Computing is the computing paradigm that promotes the idea of assembling autonomous and platform-independent software services in a loosely coupled manner for creating business processes and applications that span organizational boundaries. It presently has a dominant position in developing distributed software systems [1–4]. Web services are one of the major technologies for implementing Service-Oriented Computing, and Web Services Business Process Execution Language (WS-BPEL, or BPEL in short) [5,6] has become a de facto industry standard (which has been widely adopted by major IT service providers including IBM, Oracle, and SAP) to create composite service processes and applications.

One of the key research challenges in developing service-oriented applications is *dynamically adaptive processes*. As stated in [2], "*services and processes should equip themselves with adaptive service capabilities so that they can continually morph themselves to respond to environmental demands and changes without compromising operational and financial efficiencies*". By *dynamically adaptive*, we mean a process being able to change its behavior at runtime in accordance with the changes in the requirements and/or the external environment.

* Corresponding author.
E-mail address: jian.yu@aut.ac.nz (J. Yu).

http://dx.doi.org/10.1016/j.jcss.2014.11.008
0022-0000/

In general, changes in requirements or environment require a system to use different adaptation strategies to deal with them. One type of adaptation is *non-functional*, which means the system may need to adjust its scheduling algorithms or resources, or replace incompetent components in order to achieve the prescribed or new Quality-of-Service (QoS) attributes. Adaptation approaches proposed in [7] and [4] are examples of non-functional adaptation. The other type of system adaptation is *functional*, which requires an adaptive system to include/exclude/replace component services or change its processing logic to expose a new functional behavior. For example, a business promotion campaign may require a travel booking system to include free car rental services or give discount to specific itineraries. The merge/division in organizational structure may require the supporting software system to merge/divide its component services. The focus of this paper is on functional adaptation.

Although BPEL has been around for one decade (submitted to OASIS for standardization in April 2003), it is inherently static and has limited dynamic adaptation features [4]. BPEL supports dynamic binding of component services through partner links. However, the process definition itself would contain a large amount of code that is not related to the business process. Since then, a number of approaches (e.g., AO4BPEL [8] and VieDAME [4]) have been proposed to cope with this issue. Unfortunately, most of them are implementation level techniques that apply at the code development and execution stage.

From the software design perspective, we see the root of BPEL's incapability in dealing with dynamic adaptation as a problem caused by software developers trying to satisfy requirements using *only* the process-oriented solution. Although human problem solving is innately procedural [9], and business processes are the main tool for capturing procedures or *flow logic* in business requirements, there are also business rules, policies, and regulations[1] in the requirements which represent the *decision* aspect, and they usually much tend to change than procedures. Just as Ross [10] stated, "*the most significant changes do not come from re-engineering workflow, but from rethinking rules*". If we still use a process language to capture business rules, they are translated to imperative conditionals and branches in the process. Such an approach brings two major problems: i) any change to business rules requires the whole process to change, and ii) the traceability of business rules is usually lost in the translation.

In this paper, we propose a novel approach called MoDAR (Model-Driven Development of Dynamically Adaptive Service-Oriented Systems with Aspects and Rules) to support the systematic development of dynamically adaptive BPEL-based service-oriented systems. MoDAR adopts Model-Driven Development methodology (MDD) [11], in which software systems abstraction is specified in platform independent models (PIMs), and then the PIMs are (semi)automatically transformed into platform specific models (PSMs) using dedicated transformation tools. The MoDAR PIM is composed of three models: a *base model*, a *variable model*, and a *weave model*. The base model captures the flow logic of the requirements. The variable model captures the decision logic of the requirements. The weave model weaves the base model and variable model together using an aspect-oriented mechanism. The MoDAR PSM integrates BPEL processes and Drools[2] rules, which is dynamically adaptive in the sense that rules, representing the decision logic, can be changed at runtime without redeploying the BPEL process. Along with the methodology, we have developed an intuitive graphical development environment and a prototype execution environment. A real-life case study in the health service domain has been conducted to demonstrate the applicability of our approach. Performance evaluation on the execution environment was also conducted to demonstrate its efficiency.

A preliminary result of this study has been published in [12]. This article is a comprehensive summary of the MoDAR project and adds the following original major contributions:

- A formally defined base model language and its *variable activity* concept that supersede the previous immature *variable point* concept,
- A formally defined variable model with an ontology-based rule language,
- A detailed report of our substantial new development to the MoDAR platform, and
- An extensive evaluation of the MoDAR approach, including a performance study, a real life case study in the healthcare industry, and an in-lab usability study.

The rest of the paper is organized as follows. Section 2 briefly reviews the concept of business rules and a classification scheme of business rules, together with a motivating scenario that will be referred to throughout the paper to illustrate our approach. Section 3 is dedicated to the MoDAR development technique. Section 4 introduces the MoDAR development platform and the execution environment, along with the transformation between the models and the executable code. Section 5 evaluates our approach through detailed in-lab performance experiments and usability study. Finally, Section 6 surveys the state of the art and Section 7 concludes the paper.

2. Background

In this section, we review the concept of business rules and present a motivating scenario that will be used throughout the paper as an example.

[1] In the rest of this paper, we call them *business rules* in general.
[2] http://www.jboss.org/drools/.