The Journal of Systems and Software 84 (2011) 510–524

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Self-adapting workflow reconfiguration

R. Baird*, N. Jorgenson, R. Gamble

Department of Computer Science, University of Tulsa, 800 S. Tucker Drive, Tulsa, OK 74104, USA

ARTICLE INFO

Article history:
Received 17 December 2009
Received in revised form
19 September 2010
Accepted 21 October 2010
Available online 18 November 2010

Keywords:
Workflows
Reconfiguration
Web services
User preference
Dynamism

ABSTRACT

Because web services are highly interoperable, they are capable of providing uniform access to underlying technologies, allowing developers to choose between competing services. Workflow languages, such as BPEL, compose and sequence Web service invocations resulting in meaningful, and sometimes, repeated tasks. Their prevalence means there may be multiple Web services that perform the same operation with some better than others depending on the situation. Their potential for being unavailable at critical workflow execution times forces a reliance on such redundant services. One remedy for unavailability and situational awareness constraints is using quality of service factors and user-directed preferences to assign priorities to workflows and services to perform run-time replacement. In this paper we describe a novel approach to self-adapting workflow reconfiguration. We discuss the implementation of our approach embodied by the Next-generation Workflow Toolkit that supports runtime workflow reconfiguration using BPEL with a commercial workflow engine. A key design feature is the decoupling of user-directed changes regarding service priority from the actual workflow execution, allowing NeWT to effectively manage and recover from workflow changes at any time. We evaluate NeWT by comparing the same example across multiple commercial systems that claim reconfiguration capabilities.

1. Introduction

Many businesses today have adopted service-oriented architectures of Web services (WSs) as a development choice for their composite systems. Workflow languages can compose and sequence WS invocations to achieve meaningful task results. WSs often have competing counterpart services with equivalent functionality. Thus, applications that use workflows to invoke WSs have the potential to be fault-tolerant by incorporating interchangeable services in workflow compositions. However, the potential for an adaptive runtime response does not meet reality when WS providers change service availability, location, or version information without warning. Similarly, user-directed changes have previously been performed only after workflows have reached quiescence. Often, these changes require rewrites to bring the workflow up to current standards. The problems increase when workflows that combine procedural rules and business logic with specific service choices (Workflow Management Coalition, 2002) have no inherent mechanisms to take advantage of WS substitutions, should dynamic changes warrant it. Specific workflow dynamism challenges include

- process failure when expecting a response from a service (internal error, slow response time, or being offline),
- adapting to the best (performing, cost-effective, functionally matched) service available, and
- integrating replacement services (internally developed services, competing vendors) at runtime as business rules change.

A better approach is for the workflows to be self-adapting even as runtime changes and user updates occur. The contributions of this paper include the development of a novel workflow management framework that interfaces with a commercial workflow engine to allow runtime workflow self-adaptation, while organizing, controlling and affecting user preferences and service attributes on workflow configuration. The approach takes user-directed and environmental status mechanisms that dictate service priorities, scenarios, and situations that workflows execute under and decouples them from the actual execution of the workflows. This decoupling allows anytime changes at the user and environment level to lead to on-demand execution of workflows. The result is runtime self-adaptation of workflows, increasing their fault tolerance within their domain of execution.

We partition workflow adaptation into two complementary types: workflow definition reconfiguration and workflow instance reconfiguration. Workflow definition reconfiguration applies user-directed rules derived from QoS constraints and environmental influences to represent situational awareness in workflow specifications to reflect changes in service preferences, priorities, and availability. As an example, assume that a default workflow definition is a sequence of services chosen from three abstract service types. Being of the same type, means that they essentially perform

* Corresponding author.
E-mail address: robert-baird@utulsa.edu (R. Baird).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.10.026

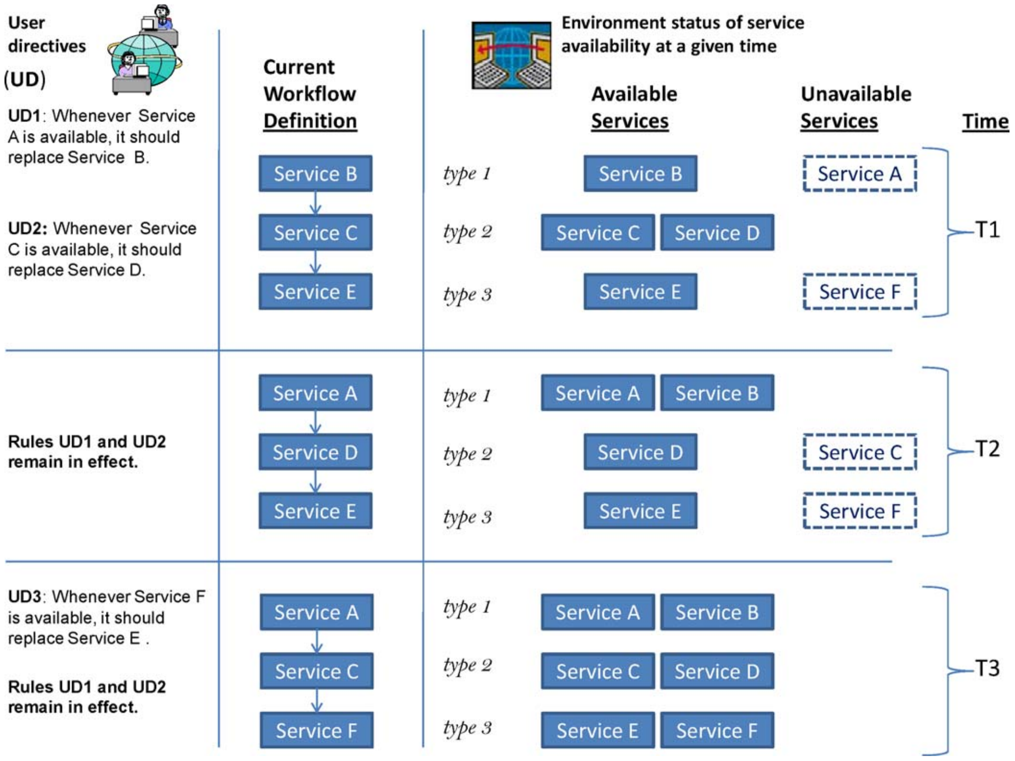
User directives (UD)
UD1: Whenever Service A is available, it should replace Service B.
UD2: Whenever Service C is available, it should replace Service D.
Rules UD1 and UD2 remain in effect.
UD3: Whenever Service F is available, it should replace Service E .
Rules UD1 and UD2 remain in effect.
Current Workflow Definition
Environment status of service availability at a given time
Available Services
Unavailable Services
Time
type 1
type 2
type 3
Service A
Service B
Service C
Service D
Service E
Service F
T1
T2
T3

Fig. 1. Workflow definition reconfiguration.

the same abstract task. Services A and B are of type 1. Services C and D are of type 2. Services E and F are of type 3. The right hand side of Fig. 1 shows the two user directed rules governing the workflow of services from the three types at time T1. The left hand side shows the current environmental status with respect to service availability and unavailability also at time T1. Given the state at T1, the current workflow definition appears as Service B (since A is unavailable), followed by Service C (since C is available) and Service E (since F is unavailable). At time T2, no new user-directed rules appear and rules UD1 and UD2 remain. Service A becomes available, while Service C becomes unavailable, causing the workflow to adapt its definition as shown at T2. At time T3, a new user directive is asserted making Service F a priority over Service E and since F has become available, the workflow definition adapts to both the user and environment.

Workflow instance reconfiguration causes executing instances to adapt to the active workflow definition at runtime if they have not yet reached the point where the definitional change was made. This adaption minimizes execution failure while attempting to produce the result expected given the user and environmental influences.

We refer to interested persons and service vendors associated with a domain for workflow development, deployment, and use as a community of interest (COI) (Renner, 2001). For example, the Red Cross is part of a COI for services that process and disseminate information during a natural disaster (Baird and Gamble, 2010). A COI works to define the needed workflows, the service types, and the QoS constraints and priorities for governing service use when multiple services of the same type are available such that different services provide replacements for workflow tasks given an alternate context or change in environment. Structured discovery and user input provides one method to support situational awareness (Li et al., 2008; M'Bareck et al., 2007; Rahmani et al., 2008). Web composition algorithms that examine large sets of services to generate plans based on different QoS attributes serve as another source of priority information for the COI (Al-Helal, 2009; Erradi et al., 2006).

We overview our Next-generation Workflow Toolkit (NeWT) that relies on the COI and environment to supply workflows with the directives needed to self-adapt. NeWT provides a framework for workflow definition and instance reconfiguration by:

- encoding and changing *service priorities and preferences* during the lifetime of a workflow,
- defining workflow *scenarios* to embed reconfiguration opportunities into a default workflow based on situational awareness of the execution environment,
- workflow definition and instance reconfiguration with automatic XML-based BPEL PartnerLink management.

We present its underlying design along with select implementation details. Using a model workflow problem, we demonstrate its adaptive response against the same workflow used in commercial products claiming reconfiguration capabilities.

2. Relevant research and applications

Workflow management systems rely on the Universal Description, Discovery and Integration (UDDI) protocol as a central repository to maintain status information about deployed services (Oasis, 2004). UDDI assists in managing individual WSs as part of the environmental influence on workflows. UDDI repositories store Web Service Definition Language (WSDL) specifications that aid service discovery and availability determination. However, the UDDI repository plays no direct role in the reconfiguration process for workflows using specific WSs.

Service ranking is a prerequisite of workflow specification and reconfiguration. QoS values provide rankings according to execution cost, execution time, availability, successful execution rate, reputation, frequency, and various other indices (Ko et al.,