Supporting Evolution in Workflow Definition Languages

Sérgio Miguel Fernandes, João Cachopo, and António Rito Silva

INESC-ID/Technical University of Lisbon
Rua Alves Redol n° 9, 1000-029 Lisboa, Portugal
{Sergio.Fernandes, Joao.Cachopo, Rito.Silva}@inesc-id.pt
http://www.esw.inesc-id.pt

Abstract. Workflow definition languages are evolving rapidly. However, there is not any agreed-upon standard for such languages. In this paper we address the problem of how to develop a workflow system that is able to cope with the constant changes in these languages. We address the definition aspect of workflow systems by distinguishing between frontend and backend languages. This way, a workflow system can be developed based on the backend language whereas the frontend language can change.

1 Introduction

In its simplest form, a *Workflow System* (WfS) allows its users to describe and execute a workflow. The language used to describe a workflow is a *Workflow Definition Language* (WfDL).

In this paper we address a software engineering problem: The problem of developing a WfS that supports changes in the WfDL. Moreover, it should be able to support more than one WfDL simultaneously. The solution we propose is to split the WfS into two layers: (1) a layer implementing a *Workflow Virtual Machine*, which is responsible for most of the WfS activities; and (2) a layer where the different WfDLs are handled, which is responsible for making the mapping between each WfDL and the Workflow Virtual Machine.

In the next section, we give some motivation for this work and discuss some of the problems that a WfS faces when it is not built with WfDL change in mind. In Sect. 3 we describe our proposal of creating a Workflow Virtual Machine to support different WfDLs, and in Sect. 4 we describe the object-oriented framework we implemented using this approach. Then, in Sect. 5 we discuss some related work and open issues. Finally, in Sect. 6 we conclude the paper.

2 Motivation

There are many WfSs available, each one of them with its own WfDL – different from all the others. Actually, despite the efforts of the Workflow Management Coalition [5], different languages give different names to the same concept (e.g.

some call “Activity” to what others call “Step”). The lack of a common language to describe workflows makes impossible the easy migration from one WfS to another. This is a problem for users – those that want to model and execute workflows.

Some authors have already recognised this problem [9] and presented a critical evaluation of some of these languages. This problem would be solved if we had a standard WfDL that all systems supported. As a matter of fact, different groups presented their proposals for such a standard [2], [3]. Yet, so far, the workflow community has not reached an agreement on which language to adopt.[1]

This proliferation of languages has disadvantages also for the developers of WfSs. It poses the question of which language to support in the system under development. If the chosen language becomes the standard, fine, but what if the chosen language is abandoned?

Even if a consensus was reached, the agreed upon language would be changing over time, as the specifications themselves are not yet stable, containing many open issues, as their own creators acknowledge [3], [2].

The WfS exists to support the execution of workflows, which are described by a WfDL. That WfDL has a certain set of concepts and constructs. Naturally, the WfS has to deal with those concepts and constructs to perform its tasks (such as the execution of a workflow). If the WfDL changes significantly (e.g., because it was abandoned and replaced by another), the WfS developers may have to rewrite a significant part of the system. Therefore, a major concern for the WfS developers is how to make it able to cope with changes in the WfDL, with the minimum impact in the rest of the system.

3 The Workflow Virtual Machine

To isolate the core of the WfS from changes to the WfDL, we propose the creation of a *Workflow Virtual Machine* (WfVM).

Although there are many workflow definition languages, they all share a common subset of concepts such as “activities,” “activity sequencing,” “data handling,” and “invoking domain-specific logic” [5]. The idea of the WfVM is that it should support these core concepts, allowing the definition and the execution of workflows described in a simple language that uses those concepts. We call this language the *backend language*. This is not the language that WfS users use to describe their workflows. That language – the WfDL we have been talking about – is what we call the *frontend language*.

With a WfVM in place, the description of a workflow in the frontend language is compiled into a workflow described in the backend language and then executed. This way, the core of the WfS does not depend on the frontend language, which can change freely – provided that it can always be compiled to an appropriate backend description.

[1] Unfortunately, we cannot foresee in a near future such an agreement.