An Interface for the Execution of Distributed Hierarchical Workflows

Daniel Beer[1], Raphael Kunis[1], Gudula Rünger
Chemnitz University of Technology, Department of Computer Science
09107 Chemnitz, Germany
E-mail: {dbeer,krap,ruenger}@informatik.tu-chemnitz.de

1. Introduction

The usage of business process software enables companies to perform their increasing amount of work in a more effective, transparent, and flexible way. Additionally, distributing the business workflows results in the possibility to both interact with external companies and balance the load. However, there is a lack of support for the distributed execution of workflows in free open-source workflow management systems (WfMSs).

In this paper we present a concept and implementation of an interface to support distributed workflow execution based on the standards defined and proposed by the Workflow Management Coalition (WfMC). In particular, these standards are the Workflow Reference Model [1], the Interoperability Abstract Specification [2], Wf-XML [7], and XPDL [3].

The objective of our work is to enable the automation of distributed workflow execution. Workflow designers should be able to provide their workflow processes to be requested for execution by a remote WfMS just by defining additional interfaces. Our approach enables designers of business processes to include the external process(es) as part of their overall process. Hence, the design of large hierarchical workflow process definitions is much easier by using the presented interface.

The contribution of this paper is the definition and realization of a lightweight interface derived from an open standard protocol that can be used for the automated communication of workflow management systems to facilitate the execution of distributed hierarchical workflow processes.

2. Interoperability model

Our current approach requires to model external workflow processes as subprocesses of a parent process. This is achieved by a hierarchical definition of workflows. Two processing models defined in the Interoperability Abstract Specification [2] have to be considered: (a) chained processes (model one) and (b) nested subprocesses (model two). Two processes are chained, if a parent process initializes and starts the child process and then continues its own execution taking no further interest in the execution of the child process. In contrast to model one the parent process in model two does not continue its execution until the child process is completed and the result is returned. Examples of the models are shown in Figures 1 and 2.

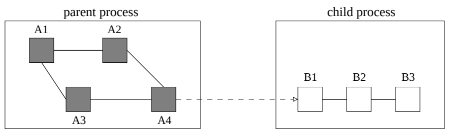

Figure 1. Chained processes (model one)

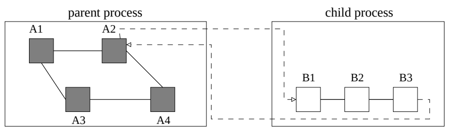

Figure 2. Nested subprocesses (model two)

3. Architecture

3.1. General overview

Our concept is based on a client/server architecture that uses asynchronous communication. This is necessary because the result of remote procedure calls (i.e. the execution of a child process) may return after a long period of time (up to months). For this purpose the concept of the Asynchronous Service Access Protocol [6] (ASAP) is adapted. ASAP defines a scheme how to communicate with services that run asynchronously by utilizing three main components (Observer, Factory, and Instance) that had to be redesigned to be suitable for workflow execution. An

[1]supported by "Bundesministerium für Bildung und Forschung" (BMBF), the German Ministry for Education and Research, project RAfEG (reference architecture for e-Government)

overview of these three components as well as an additional Application Agent, is shown in Figure 3. In our concept the parent process is executed by the client system WfMS1 and the child process is executed by the server system WfMS2.

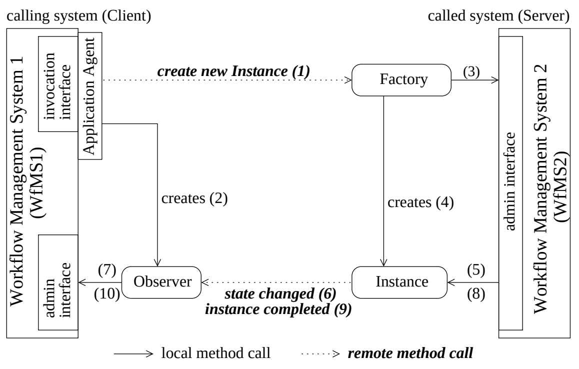

Figure 3. Overview of the main components.

The interface requires that the participating workflow management systems implement the interfaces for administration and application invocation defined in the Workflow Reference Model.

3.2. Main components

- *The Application Agent:* The Application Agent is a client object which requests the execution of the child process. It is invoked by an activity inside the parent process. The request is sent (step (1)) to a Factory server object that will be described later. The context (input and output data of the external process) is also provided by the activity that invokes the Application Agent. After the start of the child process an Observer is created that waits for information about state changes of the child process (step (2)).
- *The Factory:* The Factory server object represents a workflow process that can be executed as a nested subprocess or chained process. After the start a Factory server object waits for requests from Application Agents to create, instantiate and start the execution of the child process by WfMS2 (step (3)). In the case of a chained process the Factory processes only step (3) whereas a nested subprocess additionally requires the creation of an Instance object (step (4)).
- *The Instance:* The Instance object is the client object that informs the Observer about events during the execution of a nested subprocess, i.e. suspend, resume, complete. The Instance receives information on state change events in step (5) and the result of the child process in step (8) from WfMS2. The events are signalled to the Observer in steps (6) and (9). The data sent in step (6) comprise the new and the old state. Every nested subprocess is associated with exactly one newly created Instance. The Instance is terminated after the successful completion of the subprocess (signaled by WfMS2 in step (8)) and the transmission of the return values to the Observer (step (9)).
- *The Observer:* The Observer is a server object that waits for the transmission of information about events related to a nested subprocess. Information on state changes are forwarded to the parent process executed by WfMS1 in step (7) and the return values are forwarded in step (10). A child process has its own Observer object when it is a nested subprocess.

4. Implementation

Our current implementation uses Java as programming language and RMI as Middleware solution. The first implementation uses Enhydra Shark [5], but it is planned to extend the interface to work with other WfMSs. To store information and data on state changes we use an SQL-based database. The storing of information is especially necessary when one of the server objects is temporarily unreachable and we are able to prevent failures caused by this.

5. Application area

Our primary application area is the administration of business process execution inside and between authorities [4] where we deal with special procedures, i.e. the official approval of building plans. We are developing a software system that provides support for the distributed execution of these procedures. The procedures contain strictly defined guidelines that declare which authorities have to execute which part of the procedure and the order of the execution of the associated workflow process. Because of the participation of separated authorities (spatial as well as structural) it is mandatory that the involved workflow management systems are able to cooperate and to fulfill the requirement of a distributed workflow execution.

References

[1] D. Hollingsworth. *The Workflow Reference Model.* WfMC, Jan 1995.
[2] M. Marin. *Workflow Standard - Interoperability Abstract Specification.* WfMC, Nov 1999.
[3] R. Norin. *Workflow Process Definition Interface – XML Process Definition Language.* WfMC, Oct 2002.
[4] reference architecture for e-Government (RAfEG) http://www.rafeg.de.
[5] Shark 1.1 documentation http://shark.objectweb.org/doc/1.1/index.html, 2005.
[6] K. Swenson, J. Ricker, and M. Krishnan. *Asynchronous Service Access Protocol (ASAP).* OASIS, Jun 2004.
[7] K. D. Swenson, M. D. Gilger, and S. Predhan. *Wf-XML 2.0 - XML Based Protocol for Run-Time Integration of Process Engines.* WfMC, Nov 2004.