Chapter 72
Flexibility for Distributed Workflows

Manfred Reichert
University of Ulm, Germany

Thomas Bauer
Daimler AG, Germany

Peter Dadam
University of Ulm, Germany

ABSTRACT

This chapter shows how flexibility can be realized for distributed workflows. The capability to dynamically adapt workflow instances during runtime (e.g., to add, delete or move activities) constitutes a fundamental challenge for any workflow management system (WfMS). While there has been significant research on ad-hoc workflow changes and on related correctness issues, there exists only little work on how to provide respective runtime flexibility in an enterprise-wide context as well. Here, scalability at the presence of high loads constitutes an essential requirement, often necessitating distributed (i.e., piecewise) control of a workflow instance by different workflow servers, which should be as independent from each other as possible. This chapter presents advanced concepts and techniques for enabling ad-hoc workflow changes in a distributed WfMS as well. Our focus is on minimizing the communication costs among workflow servers, while ensuring a correct execution behavior as well as correctness of ad-hoc workflow changes at any time.

INTRODUCTION

For a variety of reasons enterprises are developing a growing interest in aligning their information systems such that they become process-aware (Lenz, 2007; Müller, 2006; Mutschler 2006; Mutschler, 2008a). Such process-aware information systems (PAISs) offer the right *tasks* at the right *point in time* to the right *actors* along with the *information*, *resources* and *application services* needed to perform these tasks (Dadam, 2000). Business process management technology offers promising perspectives to achieve this goal (Weske, 2007). Examples include workflow management systems and case handling tools (Günther, 2008 a; Mutschler, 2008b).

DOI: 10.4018/978-1-4666-4153-2.ch072

A workflow management system (WfMS) enables computer-supported business processes (i.e., *workflows*) to be executed in a distributed system environment (Bauer, 1999; Muth, 1998; Shegalov, 2001). Usually, a WfMS provides powerful tools for implementing enterprise-wide, process-aware information systems (PAISs) (Dadam, 1999). As opposed to data- or function-centered information systems, a WfMS separates the specification of the process logic (i.e., the control and data flow between the process activities) from application coding (Dadam, 2000; Weber, 2007); i.e., process logic can be described explicitly in terms of a *workflow template* providing the schema for *workflow enactment* (workflow schema for short). The different *activities*, in turn, are implemented as loosely coupled *application services* that can expect that their input parameters are provided upon invocation by the WfMS and which only have to produce correct values for their output parameters. Usually, the core of the *workflow layer* is built by the WfMS which provides generic functions for modeling, configuring, executing, and monitoring workflows.

This separation of concerns increases maintainability and reduces cost of change (Mutschler, 2008a; Weber, 2008a); i.e., changes to one layer often can be performed without affecting other layers; e.g., changing the execution order of workflow (WF) activities or adding new activities to a *WF schema* can, to a large degree, be accomplished without touching any of the associated application services (Dadam et al., 2000). Furthermore, a *WF schema* can be checked for the absence of flaws already at buildtime; i.e., deadlocks, livelocks and faulty data flow specifications (van der Aalst, 2000; Reichert, 1998a) can be excluded in an early stage of the process lifecycle (Weber, 2009; Weber, 2006a). At run-time, new *WF instances* can be created and executed according to the underlying *WF schema*. When an activity becomes activated, a respective *work item* is assigned to the *worklists* of authorized users (which are determined based on the *actor assignment* associated with the corresponding activity). One example of such a WfMS constitutes the ADEPT system we have developed during the last years (Reichert, 2003c).

Problem Statement

A centralized WfMS shows deficits when being confronted with high loads or when supporting cross-departmental processes (Reichert, 1999; Dadam, 2000). In the ADEPT project, we have considered this by realizing a *distributed WfMS* made up of several WF servers (Bauer, 1997; Bauer, 1999; Bauer, 2003; Montagut, 2007). In this distributed variant of the ADEPT system, we allow WF designers to subdivide a WF schema into several partitions which are then controlled ”piecewise” by different WF servers in order to obtain favorable communication behavior. Note that similar approaches have been discussed in literature (Alonso, 1995; Casati, 1996; Cichocki, 2000; Dogac, 1997; Gronemann, 1999; Guth, 1998; Kochut, 2003; Muth, 1998; Schuster, 1999; Sheth, 1997; Weske, 1999).

Comparable to centralized WfMS, also a distributed WfMS needs to be flexible to cover the broad spectrum of processes we can find in today’s organizations (Bassil, 2004; Kochut, 2003; Lenz, 2007; Minor, 2007; Müller, 2006; Reichert, 1998 b). Thus, at the WF instance level it should be possible to flexibly deviate from the predefined WF schema during runtime. As reported in literature (van der Aalst, 2001a; Pesic, 2007, Reichert, 1998a; Mourào, 2007; Weber 2006a) such ad-hoc workflow changes become necessary to deal with exceptional and changing situations. Within the ADEPT project we developed an advanced technology for the support of such ad-hoc changes (Reichert, 1998a; Reichert, 2003a; Reichert, 2003b). In particular, ADEPT allows authorized users (or agents) to dynamically modify running WF instances, but without causing run-time errors or inconsistencies in the sequel (Rinderle, 2003).