Process Evolution in a Distributed Process Execution Environment

Pieter Hens, Department of Decision Sciences and Information Management, KU Leuven, Leuven, Belgium
Monique Snoeck, Department of Decision Sciences and Information Management, KU Leuven, Leuven, Belgium
Geert Poels, Department of Management Information and Operations Management, Universiteit Gent, Ghent, Belgium
Manu De Backer, Department of Decision Sciences and Information Management, KU Leuven, Leuven, Belgium, Department of Management Information and Operations Management, Universiteit Gent, Ghent, Belgium, & Department of Management Information Systems, Universiteit Antwerpen, Antwerp, Belgium

ABSTRACT

To allow the distribution of control and visibility of cross-organizational process models and increase availability and performance of the processes, a process model can be fragmented into logically different parts and distributed in the enterprise architecture. Fragmentation algorithms and execution environments which connect the fragmented process model parts together, recreating the original process execution semantics, have been proposed in earlier work. However, a critical challenge that is left open is the ability of the distributed process execution environment to respond effectively to process changes. In this paper, the authors describe the difficulties, advantages and issues of process model change support in a fragmented and distributed environment. Moreover, the authors propose a system which tackles the identified issues and allows the propagation and coordination of process changes at runtime in the distributed process execution architecture.

Keywords: Business Process Enactment, Distributed Business Processes, Process Evolution, Service Oriented Architecture, Workflow Change

INTRODUCTION

Business process model fragmentation is the process of splitting a process model that was modeled as a whole into logically different, smaller model fragments with the intention to distribute the fragments over different execution and controlling partners. There are several reasons for process model fragmentation: distribution of ownership and/or coordination across process model fragments; elimination of a single point of failure during process model execution;
and increasing availability and performance of the process model execution.

Process model fragmentation allows for the distribution of control and responsibility of the process model. In contrary, using a central execution scheme to operate the complete process flow implies that the responsibility of the entire process execution lies with one organizational entity. However, it is not uncommon that processes are cross-departmental or even cross-organizational, where it is not viable that one single entity has full control over the entire process flow, or even has visibility of the entire process flow. Also, the process may be designed centrally as one unit, but off-shoring and outsourcing process capabilities may require the fragmentation of this process model.

Besides such organizational reasons, executable business process models, i.e. a process model described in an executable process language like BPEL (OASIS, 2007) or YAWL (van der Aalst & ter Hofstede, 2005) may also have technical reasons for their distribution and fragmentation. When the model is executed as one unit by one process engine (centralized process execution) and at high loads (i.e. increasing client requests), the engine has to handle a significant amount of process instances simultaneously. For complex processes this requires handling a vast state space, performing complex data transformations and invoking multiple component services (e.g. web services and task managers). This puts a high pressure on the central process engine and performance degrades as the number of process instances increases (Chafle, Chandra, Mann, & Nanda, 2004). Alongside degradation of the performance, centralized execution also adds a single point of failure to the process execution architecture. Services (e.g. web services) are distributed and decentralized, but the decision logic and coordination (composition) of these services is still located at one point (i.e. the process engine). Failure of the coordinator means failure of the entire process, even if the services themselves are still available and ready to be executed (Chafle et al., 2004; Muth, Wodtke, Weissenfels, Dittrich, & Weikum, 1998).

Process model fragmentation and distributed execution addresses these issues.

Many different techniques for process model fragmentation have been proposed in literature. For example, Chafle et al. (2004) use program dependency graphs, a tool borrowed from compiler optimization, to split up the process flow. Their goal is to reduce the network traffic involved. For the same reasons, Fdhila et al. (2009) fragment the process flow using dependency tables. To increase availability and failure-resilience Muth et al. (1998) perform process model fragmentation using state and activity charts and Khalaf et al. (2008) fragment a BPEL flow according to predefined swimlanes to enable distribution of ownership and coordination. The proposed techniques differ in the way and reasons processes are fragmented, but the result is, however, always the same: a set of logically different fragments distilled in an (automated) way from the original process model, which enables the distributed execution of each fragment by different process partners.

A problem that is left open is how process evolution and process model change is performed in these distributed process execution environments. A critical challenge for any process-aware information system (PAIS) is its ability to respond effectively to process changes (Weber, Sadiq, & Reichert, 2009; van der Aalst, 2001; Rinderle, Reichert, & Dadam, 2004). Processes evolve over time and the execution environment should adequately support these changes. The distributed execution environment adds, however, additional difficulties in process change support: a global process overview is unavailable since execution is fragmented, instances are created for fragments and not for the global process model and extra overhead is introduced since coordination between physically distributed fragments is needed to propagate changes in the execution environment.

In this paper we describe the difficulties, advantages and issues of process evolution in a fragmented and distributed process model execution environment and propose a system
Reusing Services through Context-Aware Discovery and Adaptation in Pervasive Systems
[www.igi-global.com/chapter/reusing-services-through-context-aware/77787?camid=4v1a](www.igi-global.com/chapter/reusing-services-through-context-aware/77787?camid=4v1a)

Riki: A System for Knowledge Transfer and Reuse in Software Engineering Projects
[www.igi-global.com/chapter/riki-system-knowledge-transfer-reuse/29426?camid=4v1a](www.igi-global.com/chapter/riki-system-knowledge-transfer-reuse/29426?camid=4v1a)