Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Systems Architecture 50 (2004) 309–324

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Developing a reusable workflow engine

Diogo M.R. Ferreira [a,*], J.J. Pinto Ferreira [b]

[a] *INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal*
[b] *Faculty of Engineering U.P., Rua Dr. Roberto Frias, 4200-465 Porto, Portugal*

Received 14 February 2002; received in revised form 30 July 2003; accepted 24 September 2003
Available online 27 November 2003

Abstract

Every time a workflow solution is conceived there is a large amount of functionality that is eventually reinvented and redeveloped from scratch. Workflow management systems from academia to the commercial arena exhibit a myriad of approaches having as much in common as in contrast with each other. Efforts in standardizing a workflow reference model and the gradual endorsement of those standards have also not precluded developers from designing workflow systems tailored to specific user needs. This article is written in the belief that an appropriate set of common workflow functionality can be abstracted and reused in forthcoming systems or embedded in applications intended to become workflow-enabled. Specific requirements and a prototype implementation of such functionality, named Workflow Kernel, are discussed.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Workflow management systems; Workflow engines; Component reuse

1. Introduction

The workflow reference model proposed by the workflow management coalition [17] defines a framework for relating workflow management systems and their capabilities. Within this framework, supporting tools, execution services, client applications and external applications interact according to a set of interfaces. At the heart of this framework is the workflow enactment service, an execution service comprising one or more workflow engines. According to the reference model, a workflow engine is “a software service that provides the run time execution environment for a process instance” [17].

In this sense every workflow product, prototype or approach entails a workflow engine in one way or another. Interpreting a process definition, creating process instances from those definitions, and managing the execution of those instances are essential chores of every workflow management system. These capabilities represent functionality that is coded and embedded in every workflow solution. Notwithstanding, workflow systems are usually portrayed by supporting tools such as process editors or audit trail viewers. The important workflow functionality, however, is the one

[*] Corresponding author. Tel.: +351-22-209-4329; fax: +351-22-209-4050.
E-mail addresses: dmf@inescporto.pt (D.M.R. Ferreira), jjpf@fe.up.pt (J.J. Pinto Ferreira).

creating and managing the execution of process instances by iterating through individual tasks and triggering the appropriate actions.

Up to now, this functionality has been typically implemented over and over again as each workflow management system is developed. Existing standards or common views such as the ones proposed by the WfMC could lay down the guidelines for implementing workflow engines. And, to some extent, they do. But, as pointed out by [15], existing standards focus on the syntax of the reference model interfaces without clearly specifying the respective semantics and usage. Therefore, when confronted with specific user needs, developers often make use of standards according to their own interpretation. The previous authors go even further and compare the present situation with the early days of database management when, in the absence of the relational and entity-relationship models, an incongruous set of database solutions coexisted.

In this respect, this article argues, as other authors have done, that Petri net theory could become to workflow management what the relational model became to database management. It is also argued that a reasonable amount of common workflow functionality can be abstracted from existing approaches, and that from this abstraction it is possible to develop a Workflow Kernel that can be reused and embedded in workflow-enabled systems and applications, so as to prevent repeated and discordant implementations of general workflow features.

2. Common approaches and reusability

Successful workflow products such as Staffware™, InConcert™, or FlowMark® are an elaborate compound of user requirements. Throughout the years these and other leading products have been improved in order to fulfill or anticipate particular user needs. But although they share the common goal of business process integration, each product displays its own philosophy and approach. From the event-driven process chains (EPCs) of ARIS® Toolset [12] to the four-stage workflow loop of ActionWorkflow™ [8] there are several approaches to workflow management. Many workflow solutions are built on a set of constructs that are believed to be appropriate to describe business processes. Staffware™ has its own constructs for modeling business processes, InConcert™ has another set of constructs, and ARIS® and ActionWorkflow™ have their own constructs too.

Furthermore, every product provides a process editor to graphically define or modify processes, provides support for monitoring process instances, and client applications to manage individual tasks. These support tools, which are the front-end of the workflow solution, are the functionality most seen by the user, and take a significant effort to develop. Still, behind this front-end each one of these solutions contains an engine capable of interpreting the process definition language, creating process instances from process definitions, and controlling the execution of these instances. Regardless of product philosophy, what lies in the heart of each workflow solution is an engine that glues everything together and makes the desired orchestration possible.

The WfMC's workflow reference model describes the purpose of a workflow engine by a set of features that it is supposed to address. But then, if these features are well known and in fact provided by each workflow solution, why not isolate that functionality in a component that can be plugged in or embedded in every workflow application? Some proposals have been brought up as an answer to this question.

2.1. The Drala workflow engine

The Drala workflow engine [5] is an embeddable Java component that provides a comprehensive API for defining, executing and monitoring processes. It is not a workflow management system by itself; it is rather a core of workflow functionality which is intended to simplify the implementation of workflow management systems. The Drala workflow engine already includes some support tools such as a process editor, but it allows developers to replace these tools or to build additional user interfaces. Process definitions can be imported and exported in an XML format, and the