Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Systems 30 (2005) 245–275
www.elsevier.com/locate/infosys

# YAWL: yet another workflow language

W.M.P. van der Aalst[a,b,*], A.H.M. ter Hofstede[b]

[a] Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands

[b] Centre for Information Technology Innovation, Queensland University of Technology, P.O. Box 2434, Brisbane Qld 4001, Australia

Received 20 December 2002; received in revised form 4 September 2003; accepted 5 February 2004

**Abstract**

Based on a rigorous analysis of existing workflow management systems and workflow languages, a new workflow language is proposed: yet another workflow language (YAWL). To identify the differences between the various languages, we have collected a fairly complete set of workflow patterns. Based on these patterns we have evaluated several workflow products and detected considerable differences in their ability to capture control flows for non-trivial workflow processes. Languages based on Petri nets perform better when it comes to state-based workflow patterns. However, some patterns (e.g. involving multiple instances, complex synchronisations or non-local withdrawals) are not easy to map onto (high-level) Petri nets. This inspired us to develop a new language by taking Petri nets as a starting point and adding mechanisms to allow for a more direct and intuitive support of the workflow patterns identified. This paper motivates the need for such a language, specifies the semantics of the language, and shows that soundness can be verified in a compositional way. Although YAWL is intended as a complete workflow language, the focus of this paper is limited to the control-flow perspective.


*Keywords:* YAWL; Workflow languages; Petri nets; Workflow management; Workflow patterns

## 1. Introduction

Despite the efforts of the workflow management coalition (WfMC [1,2]), workflow management systems use a large variety of languages and concepts based on different paradigms. Most of the products available use a proprietary language rather than a tool-independent language. Some workflow management systems are based on Petri nets but typically add both product specific extensions and restrictions [3–5]. Other systems use a completely different mechanism. For example, IBM's MQSeries Workflow uses both active and passive threads rather than token passing [6]. The differences between the various tools are striking. One of the reasons attributed to the lack of consensus of what constitutes a workflow specification is the variety of ways in which business processes are otherwise described. The absence of a universal organisational "theory", and standard business process modelling concepts,

*Corresponding author. Department of Information and Technology, Faculty of Technology and Management, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands. Tel.: +31-40-247-4295; fax: +31-40-243-2612.

*E-mail addresses:* w.m.p.v.d.aalst@tm.tue.nl (W.M.P. van der Aalst), a.terhofstede@qut.edu.au (A.H.M. ter Hofstede).

0306-4379/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2004.02.002

it is contended, explains and ultimately justifies the major differences in workflow languages—fostering up a "horses for courses" diversity in workflow languages. What is more, the comparison of different workflow products winds up being more of a dissemination of products and less of a critique of workflow language capabilities [7].

Workflow specifications can be understood, in a broad sense, from a number of different perspectives (see [4,8]). The *control-flow* perspective (or process) perspective describes tasks and their execution ordering through different constructors, which permit flow of execution control, e.g., sequence, choice, parallelism and join synchronisation. Tasks in elementary form are atomic units of work, and in compound form modularise an execution order of a set of tasks. The *data perspective* deals with business and processing data. This perspective is layered on top of the control perspective. Business documents and other objects which flow between activities, and local variables of the workflow, qualify in effect pre- and post-conditions of task execution. The *resource perspective* provides an organisational structure anchor to the workflow in the form of human and device roles responsible for executing tasks. The *operational* perspective describes the elementary actions executed by tasks, where the actions map into underlying applications. Typically, (references to) business and workflow data are passed into and out of applications through activity-to-application interfaces, allowing manipulation of the data within applications.

The focus of this paper is on the control-flow perspective. Clearly, this provides an essential insight into a workflow specification's effectiveness. The data flow perspective rests on it, while the organisational and operational perspectives are ancillary. If workflow specifications are to be extended to meet newer processing requirements, control flow constructors require a fundamental insight and analysis. Currently, most workflow languages support the basic constructs of sequence, iteration, splits (AND and XOR) and joins (AND and XOR)—see [1,4]. However, the interpretation of even these basic constructs is not uniform and it is often unclear how more complex requirements could be supported. Indeed, vendors afford the opportunity to recommend implementation level "hacks". The result is that neither the current capabilities of workflow languages nor insight into more complex requirements of business processes is advanced [7].

We indicate requirements for workflow languages through workflow *patterns* [7,9]. As described in [10], a "pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts". Gamma et al. [11] first catalogued systematically some 23 design patterns which describe the smallest recurring interactions in object-oriented systems. The design patterns, as such, provided independence from the implementation technology and at the same time independence from the essential requirements of the domain that they were attempting to address (see also e.g. [12]).

We have collected a comprehensive set of workflow patterns to compare the functionality of 15 workflow management systems (COSA, Visual Workflow, Forté Conductor, Lotus Domino Workflow, Meteor, Mobile, MQSeries/Workflow, Staffware, Verve Workflow, I-Flow, InConcert, Changengine, SAP R/3 Workflow, Eastman, and FLOWer). The result of this evaluation reveals that (1) the expressive power of contemporary systems leaves much to be desired and (2) the systems support different patterns. Note that we do not use the term "expressiveness" in the traditional or formal sense. If one abstracts from capacity constraints, any workflow language is Turing complete. Therefore, it makes no sense to compare these languages using formal notions of expressiveness. Instead we use a more intuitive notion of expressiveness which takes the modelling effort into account. This more intuitive notion is often referred to as *suitability*. See [13] for a discussion on the distinction between formal expressiveness and suitability. In the remainder, we will use the term suitability.

In this paper, we cannot repeat the detailed arguments given in [7]. Some readers may argue that the patterns are selected subjectively. We partly agree. Since we are not aiming at formal expressiveness but at suitability, we cannot formally prove the need for each of the patterns. However, in [7] the patterns are motivated in