YAWL: Power through Patterns

Michael Adams, Stephan Clemens, Marcello La Rosa
and Arthur H.M. ter Hofstede

BPM Group, Queensland University of Technology, Australia
{mj.adams,stephan.clemens,m.larosa,a.terhofstede}@qut.edu.au

Abstract. Workflow Management Systems (WfMSs) enable the development and maintenance of workflow specifications at design time and their execution and monitoring at runtime. The open source WfMS YAWL supports the YAWL language – a formally defined language based on Petri nets which offers comprehensive support for control-flow and resource patterns. In addition, the YAWL system provides extensive support for process flexibility, in particular for process configuration, exception handling, dynamic workflow and declarative workflow. Due to its formal foundation, sophisticated verification support can also be achieved. This paper presents the YAWL system and its main applications.

1 Introduction

While workflow automation can be traced back to the seventies, it took until the mid-to-late nineties to achieve the necessary breakthroughs for it to find mainstream acceptance. A substantial number of workflow systems were developed, many using their own vendor-specific language for workflow specification. In this context the Workflow Patterns Initiative[1] emerged in 1999. Through this initiative a number of control-flow patterns were discovered, which documented well-known workflow requirements in a language-independent manner. This allowed an objective benchmarking of workflow functionality and served as guidance for language extensions and development.

Given the limited support in commercial tools for many of the patterns and facing potential criticism that realizing comprehensive support for the workflow patterns would not be feasible, the YAWL language was developed. YAWL, one of the few formally defined workflow languages, provided comprehensive support for the workflow patterns, achieved by extending Petri nets with a number of dedicated constructs.

Over time an open-source implementation of a WfMS that supported the YAWL language natively has been developed. This system, based on a service-oriented architecture, incorporates sophisticated solutions for workflow verification and flexibility support, notably for process configuration,[2] exception handling, dynamic workflow (through the Worklet Service) and declarative workflow (through the Declare Service).[3] The original patterns collection was extended

[1] www.workflowpatterns.com
[2] www.processconfiguration.com
[3] declare.sf.net

with a collection of resource patterns and comprehensive support for these patterns in the YAWL environment was recently realised.

2 The YAWL system

Some of the fundamental goals of the YAWL environment include that it should be freely available, portable, easy to use and interoperable. By releasing the environment under an open source license free availability was achieved. Portability was achieved through the use of Java^{TM} and by avoiding any operating system dependencies. Ease of use was realised by the provision of an intuitive user interface for creating and executing specifications. Finally, interoperability was supported by a service-oriented architecture, the definition of a common XML format and a set of API calls for the exchange of workflow specifications between design time and runtime environment.

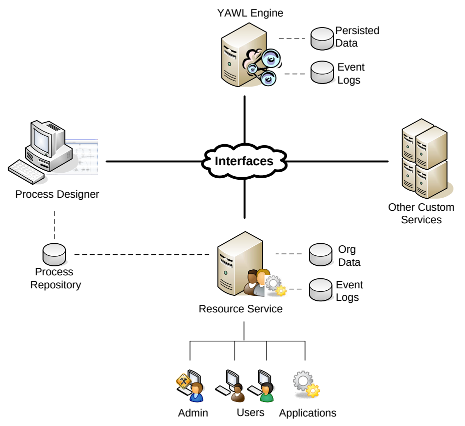
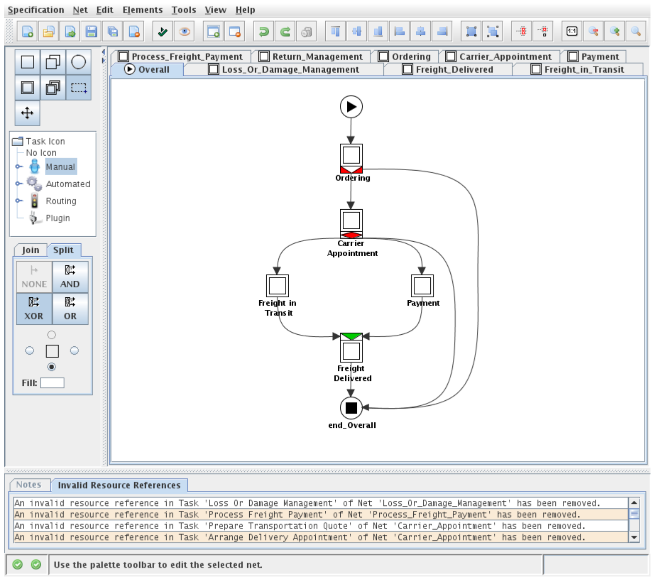


Fig. 1. Simplified architectural overview of the YAWL system.

The three main components of the YAWL system are the Editor, the Engine and the Resource Service, which communicate through well-defined interfaces (see Figure 1). They all are connected through the underlying service-oriented architecture. Based on this architecture users can adapt the YAWL system to their needs and introduce their own custom services where required.

The *Editor* provides a tool palette from which modeling elements (such as tasks or conditions) may be chosen for placement on the design canvas (see Figure 2). Routing constructs may be attached to tasks, and arcs added to link tasks and conditions, in order to form a complete workflow graph capturing a particular business process. At any time a workflow model may be verified using various algorithms to ensure completeness and soundness, amongst other things. The Editor communicates with a running Engine to receive a list of YAWL

Services which are registered with the Engine. A workflow designer can then associate those services with tasks of a workflow specification. Furthermore, from a running Resource Service, the Editor obtains organizational resources and so-called codelets. Within the Editor tasks can be associated with those resources, which are offered the according workitem at runtime. Furthermore, a workflow designer can automate a task by assigning a codelet, which is a small Java based application executed for the task at runtime.



```
An invalid resource reference in Task 'Loss Or Damage Management' of Net 'Loss_Or_Damage_Management' has been removed.
An invalid resource reference in Task 'Process Freight Payment' of Net 'Process_Freight_Payment' has been removed.
An invalid resource reference in Task 'Prepare Transportation Quote' of Net 'Carrier_Appointment' has been removed.
An invalid resource reference in Task 'Arrange Delivery Appointment' of Net 'Carrier_Appointment' has been removed.
```

Use the palette toolbar to edit the selected net.

Fig. 2. The Editor of the YAWL system.

The *Engine* deals with control-flow logic and data passing, but is resource-agnostic. Valid workflow specifications are loaded into the Engine and stored in a repository from where they may be instantiated to produce cases. The Engine is in charge of the execution of cases and determines which workitems are enabled throughout the process lifecycle. Workflow data can be updated through the execution of workitems and be passed between workflow elements or e.g. be used for the evaluation of routing conditions.

The *Resource Service* ensures the correct routing of workitems to resources. It consists of the following four sub-services: A *Resource Manager*, which manages the allocation of resources to workitems; a *Worklist Handler* – a web-form based user interface that provides users with the ability to interact with and to process workitems; a *Forms Connector* to visualize tailor-made and/or dynamically generated web-forms for displaying and editing workitem data; and a *Codelet Service* that maintains and executes codelets selected for automated tasks.

3 Maturity of YAWL

The YAWL system has been downloaded from SourceForge more than 80,000 times[4]. Being open source, and thus freely available, and having a solid theoretical base, makes the YAWL environment suitable for tertiary education — at least 24 universities worldwide have used it in their teaching. The original YAWL paper[5] has attracted over 400 citations according to Google Scholar and is the second most cited paper to appear in *Information Systems* according to Scopus.

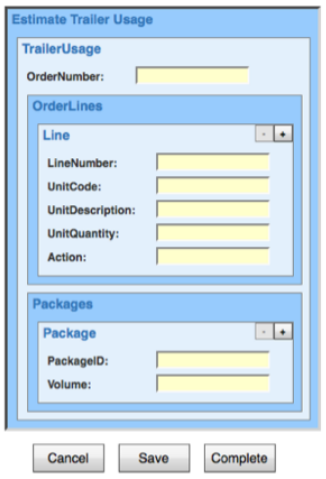


Fig. 3. A dynamically generated form.

The Australian Film, Television & Radio School (AFTRS) and the Queensland University of Technology (QUT) started the *YAWL4Film* initiative in 2007 in the context of the ARC Centre of Excellence for Creative Industries and Innovation. As part of this collaboration a solution was developed for film shoot production data management and report generation which uses YAWL. This solution was trialed in two student projects at the AFTRS and it was further adapted for the shooting of the commercial feature film “Prime Mover” in the Australian outback.[6]

first:utility and first:telecom in the UK, both part of the Impello plc group of companies, providing energy and telecoms services respectively, have been collaborating with QUT since 2005 on the YAWL initiative. These companies have built software around the YAWL system providing a novel approach to page navigation for web based systems together with the more traditional use of choreographing long-lived business processes.[7]

4 Outlook

Future work for the YAWL system is anticipated to include sophisticated workflow monitoring, conceptual process integration primitives, improved support for forms generation and templating, and system integration.

---

[4] https://sourceforge.net/projects/yawl

[5] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Information Systems, 30(4):245–275, 2005

[6] http://www.primemovermovie.com

[7] Quoted almost verbatim from http://www.yawlfoundation.org/about/adoption.html.