Proceedings of I-KNOW '07
Graz, Austria, September 5-7, 2007

Deploying YAWL for Workflows in Workplace-embedded Learning

Eicke Godehardt, Markus Doehring, Andreas Faatz, Manuel Goertz
Fraunhofer IGD, Germany
eicke.godehardt@igd.fraunhofer.de;
SAP AG, Germany
{markus.doehring, andreas.faatz, manuel.goertz}@sap.com

Abstract: In today's information society the border between the roles of learners, workers and teachers becomes more and more fuzzy. To support this, we are using the workflow language YAWL. It provides guidance in work processes and workplace embedded learning. Nevertheless, using workflows in user driven (e-learning) environments, triggers challenges. We discuss the ex-post OR-split, a novel recommendation of how to cope with OR-splits. Furthermore, the visualisation strategy on how to even simplify the graphical view of YAWL on the workflow during the workplace-embedded learning application is discussed.
Key Words: workflow, YAWL, workplace-embedded learning
Category:

1 Introduction

We are faced with a shift to the knowledge society. A great challenge for the next generation of e-learning systems is workplace-embedded learning, which brings electronically available learning material and knowledge items of any kind directly to the working user at her workplace. In such a scenario, the learning material will not be organized in curricula or courses any longer, but in relevance to the working situation. In particular this is challenging for knowledge intensive work. APOSDLE (Advanced Process-Oriented Self-Directed Learning Environment) [1, 2, 3] is a research project partially funded by the EU, which investigates and prototypes seamlessly integrating roles of teaching, (knowledge-intensive) working and learning in next-generation e-learning systems.

Our paper concentrates on aspects of melting working and learning. Today large organizations—but increasingly small and medium sized enterprises too—formalize their work by processes/workflows. Workflows are instances of business processes, i.e., of "a set of one or more linked procedures or activities, which collectively realize a business objective or policy goal, normally within the context of an organizational structure, defining functional roles and relationships"[4]. We restrict ourselves to those parts of the workflow, which encapsulate and order the working tasks, i.e., the order of steps a worker has to follow. These steps are expressed as the control flow of a workflow [5]. The more an e-learning system

would be able to reflect these real-world workflows, the more likely e-learning becomes workplace-embedded in those cases, where workflows determine a worker's day. Thus, the core questions of our paper are:

- How can we design and deploy the e-learning functionality in an arrangement, which is as similar as possible to real-world workflows?
- What is needed in terms of a workflow language to close the gap between workflows in e-learning and working?
- Which additional interpretations and language constructs are needed to apply a concrete workflow language in workplace-embedded learning?

The rest of the paper is organised as follows. Section 2 gives a brief overview on related work, which already used workflow systems as a part of e-learning systems. Section 3 describes our approach, in particular how we selected a suitable workflow language, its characteristics and which adjustments as well as additional implementations were necessary for modelling and graphical presentation at the workplace. We summarise our findings in section 4 and give an outlook on what to solve in future increments of the APOSDLE system. We will use the term APOSDLE for the project as well as for the overall technical system developed in the project.

2 Related Work

There are some related research projects in using workflows for learning. But none of them is workplace embedded or focusing on informal learning based on workflows. Cesarini et. al. [6] uses workflows to model and structure e-learning content. In a similar manner [7] tries to use the greater flexibility for structuring e-learning content to enhance SCORM based content. Another project [8] is also not workplace-embedded, but uses workflows to model traditional learning content to make it more adaptive.

Quite similar approaches are using business-process oriented knowledge management (BPOKM) methods or are process oriented in general [9]. For example the project LIP (Learning In Process) [10], which focuses more on e-Learning supported by processes rather than informal learning and do not use a dedicated process definition language but ontologies. Another project facing an analog direction is APO-Pilot [11]. They use processes mostly to model the learning activities. On the other hand APOSDLE focuses more on helping the knowledge worker to get her work done by providing resources and collaboration to experts matching the context of the user in one selected (pre)defined process.

To summarize all of these approaches do not have the goal to integrate informal learning into the workplace and thus still stick to fixed roles for learners and teachers in a dedicated e-learning environment.

3 Approach

This section describes our approach in detail. We explain, why we chose YAWL as a modelling language, introduce necessary constructs and present our visualization principles.

3.1 Choice of the workflow language and environment

The process of choosing a suitable workflow language was driven by the following core requirements: intuitive graphical notation of the language, full formal specification, expressiveness, availability, costs and extendability. An easy and intuitive graphical notation is necessary in the APOSDLE context to enable users with a non-IT background to model the workflow in a reasonable time. On the other hand, the workflow language should be formally specified, such that a workflow expressed in this language has a unique non-ambiguous interpretation. Along with the last two requirements, the high expressiveness of the language should minimize the "effort needed to construct models that reflect the process logic in a direct manner"[12] while manipulating/editing via a graphical user interface. This goes beyond usability—the concern is more on the logical expressiveness and the encapsulating constructs of the language.

The remaining requirements are non-functional. Availability is concerned about the existence of editing tools and a workflow management system, which should be deployable with reasonable costs, as the workflow modelling tool and the workflow engine are part of a larger environment, which will produce further costs itself, and allow extensions. We compared five approaches or languages (Petri Nets [13], OMG's Business Process Modeling Notation (BPMN) [14], OMG's Unified Modeling Language (UML) [14], IDS Scheer's Event Driven Process Chains (EPC) [15], ConcurTaskTrees (CTT) [16] and YAWL [17]) by hands-on evaluation and by reflecting on/studying theoretical literature [17].

Criterion	Petri Nets	BPMN	UML	EPC	CTT	YAWL
(1) Easy / Intuitive Graphical Notation	O	O	+	O	O	+
(2) Full Formal Specification	+	O	−	O	O	+
(3) High Degree of Expressiveness	−	+	O	O	O	+
(4) Available Tools and Environment	+	+	+	+	+	+
(5) Low cost	+	−	+	−	+	+
(6) Flexibility	O	O	+	−	O	+
+ accomplished | O partly accomplished | − not accomplished						

Table 1: Decision Matrix for the Selection of a Workflow Management System

The YAWL language and environment fits best the requirements. It combines a clear formal specification with an easy-to-learn and (in comparison) simple-to-use language, which can express complex workflows. Moreover, YAWL is not only a theoretical approach, but comes along with free open-source applications available at [18].

The description of all language constructs and the YAWL system is beyond the scope of this paper. The concerned reader is referred to [17] for a deeper insight. In the following section we will explain our changes and additions to YAWL.

3.2 Application of YAWL constructs

After finally choosing YAWL as our underlying workflow system, we figured out, that there is still some missing functionality according to the environment in APOSDLE or workplace-embedded e-learning in general. Especially from the fact, that human beings are using and driving the system, additional needs arose. First the user (APOSDLE knowledge worker) does not have to follow the proposed path through the workflow, she can choose every task at any time. This will lead to another obstacle—the handling of OR-splits. At start time of an OR-split the system cannot determine which tasks/paths inside the OR-split will be executed by the user. Thus this decision have to be postponed, while the original behavior of YAWL as a workflow engine was to decide at the OR-split start, which paths are chosen.

We will now show, how we applied and extended the basic YAWL constructs to fulfill the requirement defined in APOSDLE.

3.2.1 Treatment of Ex-Post OR-Splits

A significant obstacle, which has to be overcome regarding the control-flow perspective is the retroactive selection of tasks after an OR-split. As shown in figure 1, the workflow engine decides about the further routing (i.e., which tasks are subsequently enabled—tasks A and B) right at the point of time when the OR-split takes place. This is exactly after the corresponding OR-split task has finished firing. The further routing depends on circumstances like user selections or internal data variables.

This kind of process structure occurs every time the users has the ability to choose n of m parallel task until she decides to finish and go to the following task. To give an example for this kind of OR-split the overall task is to book a business trip. Usually the user has to book a flight, a car at the destination and a hotel. Nonetheless all of them are optional, she might go by train, do not need a car or stay with a friend at the business location, but the user knows which of the paths she has to take to finish the booking.

The problem is, that once a decision about the further routing has been made, it is not possible to alter it any more. Nevertheless, it is sometimes desirable, especially in user-driven environments, to allow for a more **"dynamic" decision** without requiring a complex underlying model. This is an eminent issue in the APOSDLE learning environment, because a worker rarely knows in advance what she will do next, if she has the choice. For example, if he has the options to complete his travel expense report and to work on a power-point presentation, the processing of the expense report may depend on the progress she makes with the presentation.

We conclude that the exact composition of chosen tasks should not be fixed at the beginning. It should rather be permitted to enable tasks even after the processing of the actual OR-split. We call this an "ex post or-split" or "lazy or-split". Related to our example, even if only task A and B had been selected (marked black within figure 1), task C could be retroactively selected, but only before the firing of the corresponding OR-join.

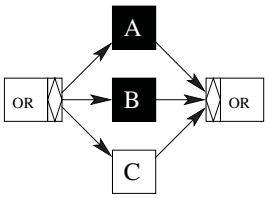

Figure 1: Problems Resulting from YAWL's OR-Split Semantics

We elaborated two different workarounds to circumvent this problem. The first one makes use of the YAWL multiple instances (MI) process element. Instead of statically defining the control-flow using an OR-split element as shown above, we replace the whole construct with multiple instances of a composite task.

The decomposition of the MI-task is made up of a XOR-split and a XOR-join, which embrace the tasks of our former OR-split-construct (tasks A, B and C in figure 2). The effect of the resulting complex is, that at the starting of the MI-task, a number of instances are created (equal to the number of branches that would have been chosen by the OR-split). Until all instances have terminated, it is possible to add new instances and therefore to enable formerly inactive tasks. Within the decomposition, the XOR-split decides which particular task has to be activated. To prevent a double activation of a task, the XOR-split stores and evaluates data within the workflow engine. If an instance would cause a double activation of a task, the XOR-Split chooses the empty task E and the instance is consequently immediately terminated.

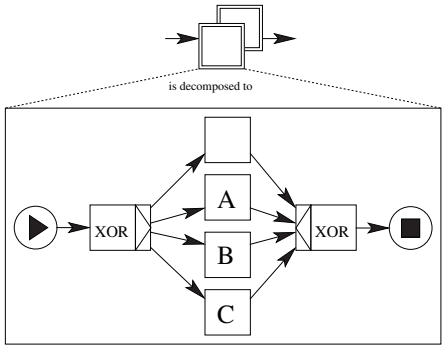

Figure 2: Solution Approach to Ex-Post OR-Split with Multiple Instances

Especially because of the double activation of tasks, we decide to use another solution approach instead. This time we realize the ex-post OR-split with the help of an initial AND-split and a cancellation region. Cancellation region is a construct from YAWL. If a task is connected to a cancellation region and the task is fired, all tasks in this region are canceled and all tokens from conditions inside the region are removed. This solution can be gathered from figure 3, where the AND-split prophylactically enables all succeeding tasks.

It is then up to the user to process and execute a selection or all of the tasks. The task complex can be finished, as soon as the explicit condition P contains a token (which again realizes the OR-join's preliminary of at least choosing and completing one branch). A characteristic of this approach is that the processing of the task complex has to be explicitly finished by the execution of the task F, which cancels all remaining active tasks and remove all tokens from conditions in the cancellation region. The cancellation of those enabled (or even running) tasks is smoothly possible. Its explicit finishing also does not constitute a critical problem within the application domain of APOSDLE. Generally, no counter-arguments against the use of this construct could be spotted. We will consequently use this **second approach** instead of the classical OR-construct in APOSDLE.

3.3 Visualizing

The intension of visualization the worker's process in APOSDLE is to give her an idea of the current task as well as the context of this task, e.g. which tasks are previous/next. On the other hand, this process visualization is also used as a user interface to select a different task, i.e., to change the worker's context.

In order to provide an intuitive user interface, we hide all special/artificial constructs from the user. Especially the joining condition P and the direct arc

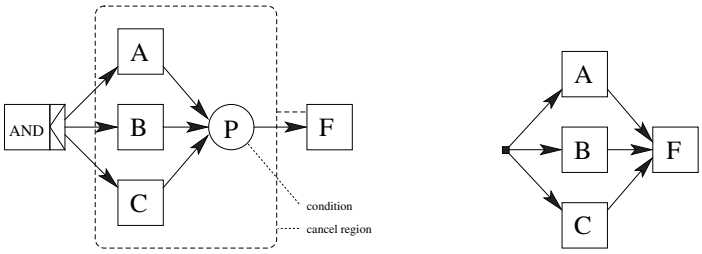

Figure 3: Ex-Post OR-Split with AND-Split and Cancellation (left side) and it's simplified visualization (right side)

to this condition would confuse regular users. Every branch just ends in the task following this condition. This is the same approach YAWL uses for implicit conditions between two tasks, which are hidden as well. The Ex-Post OR-Split solution together with the visualization concept can be seen in figure 3. As we have implemented all splits in the workflow models with our ex-post or-split, we do not have to distinguish AND- from OR-splits in the user interface. Please note that this kind of guidance is specific for the less restrictive role a workflow plays when serving as a navigation in e-learning instead of data processing.

4 Conclusion and future work

In this paper we have shown how we choose YAWL as our workflow management system, based on the requirements in the APOSDLE project. We also explained in depth the obstacles we faced and our approach by using YAWL's means to fulfill our needs and to visualize this in a way, that is easy understandable even by non-IT users.

Further research topics are the integration of desktop monitoring [19] allowing us to automatically recognize task/context changes of the user. In addition the arbitrary "jumping" in a predefined process is not completely handled yet.

Other open topics are using a different way to visualize the process, e.g., using Topic Maps [20] or using the desktop monitoring to build ad-hoc processes instead of predefine them by knowledge designer.

After showing the feasibility of our approach, it would also be interesting to transform or convert other existing workflows to YAWL and analyse their behavior applying our results.

References

1. H. Mayer, W. Haas, G. Thallinger, S. Lindstaedt, and K. Tochtermann. Aposdle - advanced process-oriented self-directed learning environment. Poster Presented

on the 2nd European Workshop on the Integration of Knowledge, Semamtic and Digital Media Technologies, 30 November – 01 December 2005.
2. S. N. Lindstaedt, T. Ley, and H. Mayer. Integrating working and learning with aposdle. In *Proceedings of the 11th Business Meeting of Forum Neue Medien, 10–11 November 2005, Vienna*. Verlag Forum Neue Medien, 2005.
3. APOSDLE. Aposdle (advanced process-oriented self-directed learning environment) - website. `http://www.aposdle.org`.
4. Workflow Management Coalition. Terminology and glossary (wfmctc-1011). `http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf`, 1999.
5. W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. In *Distributed and Parallel Databases*, volume 14, pages 5–51, Norwell, 2003. Kluwer Academic Publishers.
6. Mirko Cesarini, Mattia Monga, and Roberto Tedesco. Carrying on the e-learning process with a workflow management engine.
7. Kwang-Hoon Kim, Hyuk-Jae Yoo, and Hak-Sung Kim. A process-driven e-learning content organization model. In *ICIS '05: Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pages 328–333, Washington, DC, USA, 2005. IEEE Computer Society.
8. Peter Brusilovsky, Elmar W. Schwarz, and Gerhard Weber. A tool for developing adaptive electronic textbooks on www. In *WebNet*, 1996.
9. A. Abecker, G. Mentzas, M. Legal, S. Ntioudis, and G. Papvassiliou. Business-process oriented delivery of knowledge through domain ontologies. In *Proceedings of DEXA conference, TAKMA-2001, Second International Workshop on Theory and Applications of Knowledge Management*, 2001.
10. Andreas Schmidt. Bridging the gap between knowledge management and e-learning with context-aware corporate learning. In *Professional Knowledge Management*, pages 203–213, 2005.
11. Frank Fuchs-Kittowski, Katja Manski, Daniel Faust, Marko Prehn, and Ingo Schwenzien. Arbeitsprozessorientiertes e-learning mit methoden und werkzeugen des prozessorientierten wissensmanagement. In Arndt Bode, Jörg Desel, Sabine Rathmeyer, and Martin Wessner, editors, *DeLFI*, volume 37 of *LNI*, pages 392–401. GI, 2003.
12. W.M.P. van der Aalst. Making work flow: On the application of petri nets to business process management. In *Lecture Notes in Computer Science*, volume 2360, pages 1–22. Springer-Verlag, 2002.
13. Carl-Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.
14. OMG (Object Management Group). Website. `http://www.omg.org`, 2006.
15. G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter "Prozeßketten (EPK)". In *Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes*, volume 89, Saarbücken, 1992. Universität des Saarlandes.
16. F. Paterno, C. Mancin, and S. Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings Interact '97*, Sydney, 1997. Chapman&Hall.
17. W.M.P van der Aalst and A.H.M. ter Hofstede. Yawl - yet another workflow language. In *Information Systems*, volume 30, pages 245–275, Orlando/New York, 2005. Elsevier Ltd.
18. YAWL-Foundation. Yawl (yet another workflow language) - website. `http://www.yawlfoundation.org`.
19. Robert Lokaiczyk, Andreas Faatz, Arne Beckhaus, and Manuel Goertz. Enhancing just-in-time e-learning through machine learning on desktop context sensors, 2007. SUBMITTED.
20. TopicMaps. ISO/IEC 13250.