Supporting the Workflow Management System Development Process with YAWL

R.S. Mans[1], W.M.P. van der Aalst[1]

Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{r.s.mans,w.m.p.v.d.aalst}@tue.nl

Abstract. In order to address particular needs from the healthcare domain, the open-source YAWL Workflow Management System (WfMS) has been extended with features for scheduling support and inter-workflow support, the YAWL4Healthcare WfMS. As part of this undertaking, a WfMS development approach has been followed in which the same conceptual model is used for *specifying*, *developing*, *testing*, and *validating* the operational performance of a new system. In this paper, we elaborate on the features of YAWL that were essential for realizing our development approach.

1 Introduction

Nowadays, hospitals are investigating the introduction of a Workflow Management System (WfMS) in order to support their care processes. However, due to the complex nature of healthcare processes, these systems need to be enriched with additional functionality. Furthermore, the introduction of new technology requires a seamless integration with running operational processes and no unexpected break-downs may occur.

For ensuring above mentioned aspects, as shown at the bottom of Figure 1, we propose a WfMS development approach consisting of five phases. First, in the *requirements* phase, the required functionalities are identified. Second, during the *design* phase, a conceptual model is developed which is a formal, complete, and executable specification of the WfMS to be developed. Subsequently, the WfMS is developed in the *implementation* phase. Finally, during the *testing* and *simulation* phase, the conceptual model and the operational WfMS are used to both test and validate the operational performance of the WfMS.

The conceptual model has been defined in terms of a *Colored Petri Net* (CPN) [3]. CPNs provide a well-established and well-proven formal language suitable for describing the behavior of systems exhibiting characteristics such as concurrency, resource sharing, and synchronization. Moreover, we can use CPN Tools for specification, verification, and simulation. As concrete WfMS, the open-source *YAWL* WfMS [2] has been used. In this paper, we focus on the role of YAWL in the WfMS development approach. That is, we elaborate on the specific features of YAWL that were essential for realizing our approach.

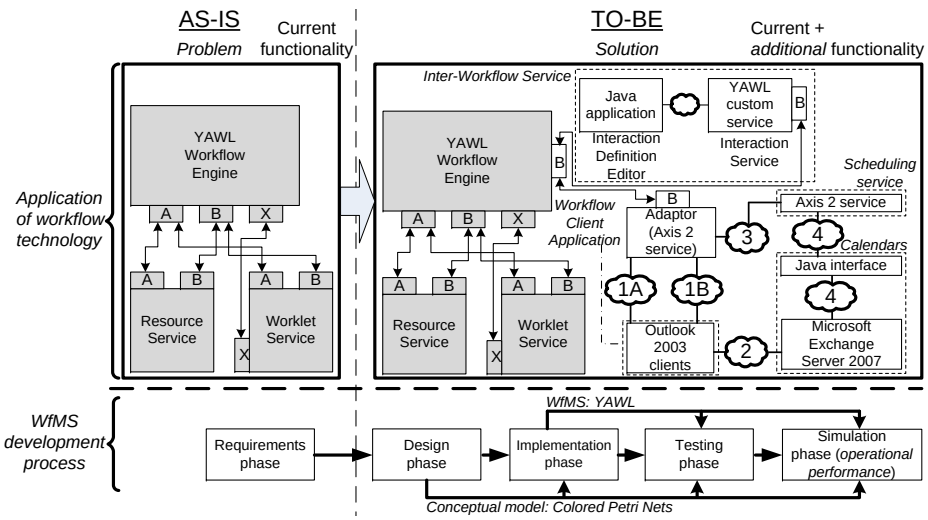


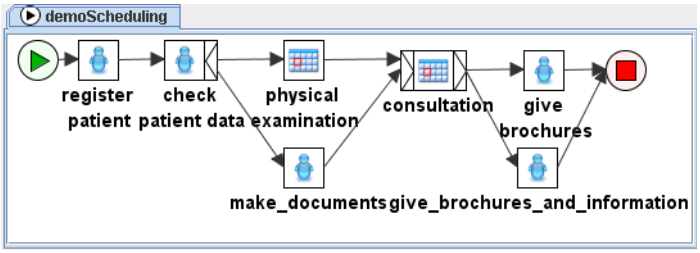
Fig. 1. Workflow Management System development approach using YAWL and the CPN conceptual model.

When applying the development approach, in the *requirements* phase it has been identified that for optimally supporting healthcare processes, WfMSs need to be extended with facilities for both *scheduling support* and *inter-workflow support* [4]. Based on the conceptual CPN model defined in the *design* phase, YAWL has been extended with the aforementioned two facilities resulting in the *YAWL4Healthcare* system and which is discussed in Section 2. Next, in Section 3 we elaborate on the *testing* and *simulation* phase in which YAWL4Healthcare is both tested and validated. Finally, we conclude in Section 4.

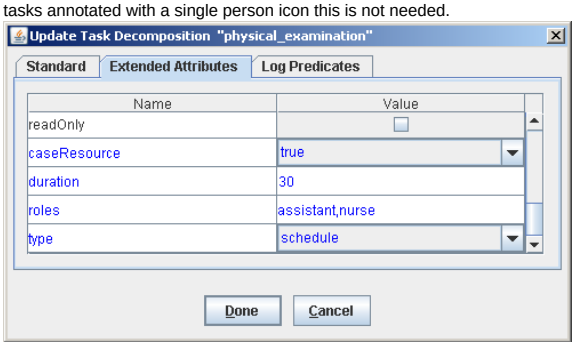
2 Scheduling Support and Inter-Workflow Support

YAWL4Healthcare offers both scheduling support and inter-workflow support. In this section, both facilities are discussed in detail and it is indicated which features of YAWL were essential for realizing them.

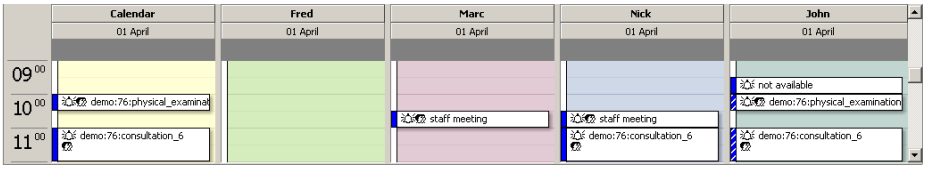
With regard to scheduling support, instead of only offering a workitem to a user via a worklist, it also needs to be possible to make an appointment for a workitem. This appointment has a specific duration and only appears in the calendars of these users that are involved in the execution of the workitem. The main scheduling features will be illustrated using a small scenario. The process definition specified in the YAWL editor can be seen in Figure 2a. The "physical examination" and "consultation" tasks are annotated with a calendar icon as for both an appointment is needed. Moreover, they are called *schedule* tasks. Via the "extended attributes" feature of the editor, additional attributes are defined for them. These attributes are also illustrated in Figure 2a in which for the "physical examination" it is defined that the presence of the patient is required during



a) Defining a model in the YAWL editor. For tasks annotated with a calendar icon an appointment is needed whereas for tasks annotated with a single person icon this is not needed.



b) The details for the 'physical examination' task are defined via the extended attributes feature of the editor.



c) State of the calendars after scheduling. The calendars of assistant 'Jane', assistant 'Fred', doctor 'Marc', doctor 'Nick', and patient 'John' are shown respectively. When scheduling the availability of resources is taken into account.

Fig. 2. Scheduling of appointments within the YAWL4Healthcare system.

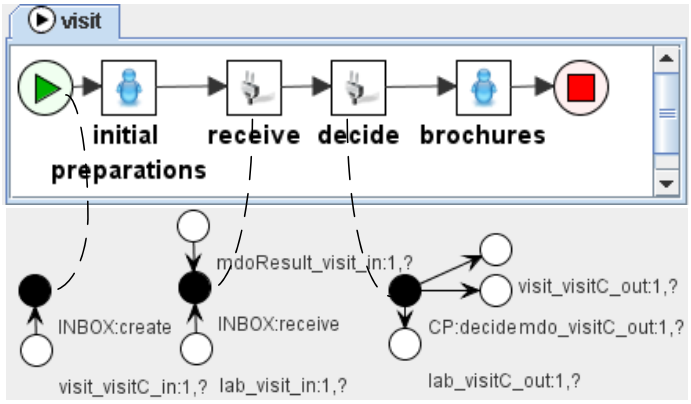
the appointment (caseResource), the average duration of the appointment is 30 minutes (duration), an assistant and a nurse are required during the appointment (roles), and the task is a schedule task (type). The tasks for which workitems need to be offered via a worktray are called *flow* tasks and are indicated by a person icon. Also, for this kind of task, additional information is defined via the "extended attributes" feature of the editor (e.g. the average duration).

Once an instance of a process is started, appointments are automatically booked in the calendars of these persons that are involved in the execution of a schedule task. This is shown in Figure 2b in which an appointment is booked for the "consultation" task which appears in the calendars of doctor "Nick" and patient "John". Also, an appointment is booked for the "physical examination". Note that the system ensures that the final scheduling of tasks occurs in the same order as the sequence of schedule tasks in the accompanying process definition for the case. Moreover, sufficient time is reserved between two scheduled tasks. In case it is found out that too little time is left for performing preced-

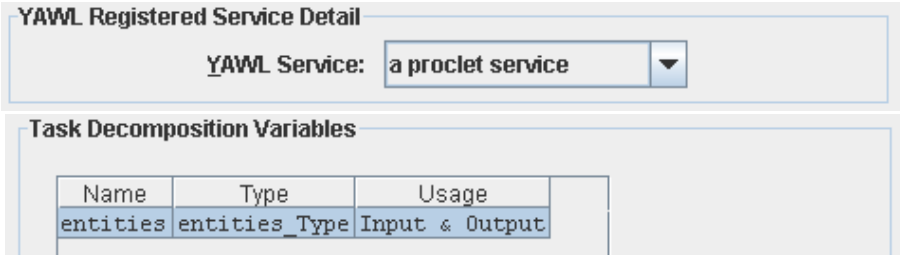
ing work-items for a scheduled schedule task, the corresponding appointment is automatically rescheduled. Also, users are able to express their dissatisfaction with the nominated scheduling by requesting: (1) the rescheduling of the appointment, (2) the rescheduling of the appointment to a specified date and time, or (3) the reassignment of the appointment to another employee.

In order to offer scheduling facilities (see Figure 1), YAWL has been extended with three components. The *Calendars* component provides a view on the calendars of users and allows for manipulating them. Based on the calendars, the *Scheduling Service* component provides the scheduling facilities to the WfMS. The workitems and any appointments for them can be seen via the *Workflow Client Application*. The "Calendars", "Scheduling Service", and "Workflow Client Application" components have respectively been realized by developing a java service, using Microsoft Exchange Server 2007, and using Outlook 2003 clients. Via a specific adaptor, communication takes place between the YAWL workflow engine and the "Scheduling Service" and the "Workflow Client Application". Moreover, in this way, only Interface B is needed for communication with the engine, i.e. starting and canceling instances, and the checking in and out of workitems. Note that the scheduling algorithm used by the Scheduling Service can easily be replaced by another algorithm, i.e., it is *pluggable*. Currently, a "naive" algorithm is implemented which searches for the first opportunity in which one of the resources of a role can be booked for the respective work-item.

The need for inter-workflow support emerged from the fact that the process of treating a patient typically consists of many smaller interacting workflow fragments that run in conjunction with each other. These fragments may also operate at different levels of granularity. The main inter-workflow support features will be illustrated using a small scenario. Note that these features are based on the Proclets framework [1] which allows for modeling and executing lightweight workflows that may interact with each other and reside at different levels of granularity. At the top of Figure 3a, the process definition of a workflow fragment, describing a visit to the hospital, is defined in the YAWL editor. At the bottom, the associated definition in the Interaction Definition Editor is shown. Via this editor it is possible to define for a workflow fragment which interactions with other fragments are necessary. That is, for tasks and input conditions for which interactions with other fragments are necessary, a so-called interaction point (visualized by a black dot) is defined. Via ports (visualized by a white dot), interaction points are connected with interaction points of other fragments. For example, for the "decide" task it is possible to instantiate a workflow fragment in order to perform a lab test or to instantiate a fragment arranging the next visit of the patient. Moreover, the patient can be registered for a multidisciplinary meeting in which the status of multiple patients is discussed. Note that at run-time for a certain entity (e.g. a patient or a lab test) an *interaction graph* is kept. This graph stores for an entity (e.g. a patient) all the interactions that need to take place between (future) fragments. As part of this, for a task for which interactions are needed, in the YAWL editor it is indicated that its exe-



a) Defining a workflow fragment and the corresponding interactions in the YAWL editor (top) and the Interaction Definition Editor (bottom). Via dotted arcs, the tasks and the interactions with other fragments that are defined for them are indicated.



b) For the 'decide' task it is indicated in the 'Task Decomposition' details that its execution is delegated to the Inter-Workflow Service. Also, the 'entities' variable allows for indicating at run-time the names of the entities for which the process is executed.

Fig. 3. Definition of interactions between workflow fragments.

cution is delegated to the Inter-Workflow service (see Figure 3b). Additionally, information about the involved entities is exchanged via the "entities" variable.

In order to offer inter-workflow support (see Figure 1), YAWL has been extended with the *Inter-Workflow Service*. The "Interaction Service" is responsible for the interactions between fragments at run-time and has been set-up as a YAWL custom service. As such, it communicates with the YAWL engine via Interface B. The "Interaction Definition Editor" offers tools for defining interactions between fragments at both design-time and run-time.

So, it can be concluded that *the service oriented architecture of YAWL was of great help for realizing the desired extensions*. That is, via the extended attributes feature of the editor, additional attributes of a task that need to be filled in, could easily be defined. Furthermore, due to the fact that the YAWL engine is agnostic to its external services, it was possible via a single interface (Interface

B) to focus only on implementing the functionalities that were needed for the desired scheduling support and inter-workflow support. So, we did not need to bother about basic workflow management functionalities as they were already provided by YAWL itself (e.g. a workflow engine and process editor).

3 Replacement of Functionalities

As illustrated in Figure 1, the conceptual model developed using CPN Tools has been used for both testing and validating the operational performance of the implemented YAWL4Healthcare WfMS in respectively the *testing* and *simulation* phase. In Figure 4 it is schematically depicted how this has been realized.

The CPN conceptual model provides a complete, formal description of the functionality of the system. As this model is executable, it also serves as a prototype implementation of the system. So, in the *testing* phase, one or more parts of the CPN model can be *replaced* by the concrete implementation of these parts. So, in the figure, the light grey-colored rectangle represents the parts of the actual system that are tested. This is done by establishing connections between the CPN model and parts of the actual YAWL4Healthcare WfMS which need to be tested. During the testing, parts of the system, which are not tested, may be simulated using CPN Tools (the dark grey-colored rectangle). This allows for the testing of numerous scenarios facilitating the discovery of potential flaws in both the architecture and the corresponding implementation.

In the *simulation* phase, our focus is on investigating the operational performance of processes that are supported by both the *designed* and *realized* system. Here we can benefit from the fact that CPN models can also be used for simulation-based performance analysis in order to evaluate systems and to compare alternative configurations of a system [3]. So, for processes supported by our system, it can be investigated whether they are negatively impacted or not (e.g. introduction of bottlenecks into the process). By setting up a reference environment, the operational performance can be investigated in a structured manner. Moreover, parts of the realized YAWL4Healthcare WfMS can be included in the CPN simulation model in order to analyze the implemented system.

At the time the above mentioned approach was applied, the "Inter-Workflow Service" was not developed yet. So, in the conceptual model, we only had com-

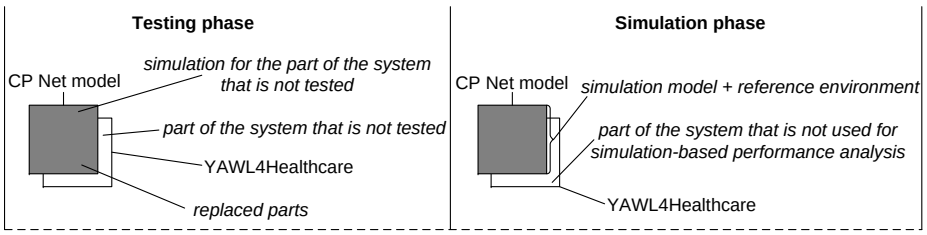


Fig. 4. The role of the CPN conceptual model and the YAWL WfMS in the testing and simulation phase.

ponents for the "Workflow Engine", "Workflow Client Application", "Calendars", and "Scheduling Service". In the testing phase, we could easily replace the scheduling service and both the workflow engine and scheduling service by their implemented YAWL4Healthcare counterparts. Furthermore, the engine was provided with a simple process definition. Also, four users where created in the "Workflow Client Application" together with the associated empty calendars in the "Calendars". Amongst others, 6 errors were identified in the "Scheduling Service" component and 6 errors were identified in the adaptor component. In the simulation phase, also both the "Workflow Engine" and "Scheduling Service" were replaced by their implemented counterparts. Furthermore, the engine was provided with a gynecological healthcare process such that a series of simulation experiments could be carried out in which 143 patients followed the process. For the schedule tasks, the corresponding appointments were made by our system. Therefore, the calendars of the simulated system were filled with appointments, etc. to mimic the true availability of the medical staff. As a result it was possible to explore different scenarios. For example, in one of the experiments we simulated the situation that the appointments for a CT, MRI, and pre-assessment are scheduled on the same day. This resulted in a considerably increased waiting time for both the pre-assessment and examination under anesthetic appointments.

Similarly, as in Section 2, also here we benefited from the service-oriented architecture of the YAWL WfMS. Due to Interface B, components in the conceptual model could easily be replaced by their implemented counterparts.

4 Conclusion

In this paper, we focussed on the role of YAWL in a development approach in which the same conceptual model is used during the design, implementation, testing, and simulation phase. By applying this approach, YAWL has been extended with facilities for scheduling support and inter-workflow support.

As part of this undertaking, we greatly benefited from the service oriented architecture of YAWL. As it turned out, this provided a powerful point of extensibility, allowing for the extension of the engine with additional desired functionalities. Moreover, it allowed for the applicability of our development approach in which there is a tight coupling between the conceptual model and the YAWL4Healthcare WfMS. While testing and validating, parts of the system may be simulated while connected to the actual system components.

References

1. W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.
2. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell, editors. *Modern Business Process Automation: YAWL and its Support Environment*. Springer-Verlag, 2010.

3. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4):213–254, 2007.
4. R.S. Mans. *Workflow Support for the Healthcare Domain*. PhD thesis, Eindhoven University of Technology, June 2011. See http://www.processmining.org/blogs/pub2011/workflow_support_for_the_healthcare_domain.