Supporting Healthcare Processes with YAWL4Healthcare

Ronny S. Mans[1,3], Nick C. Russell[2], Wil M.P. van der Aalst[1], Arnold J. Moleman[3], Piet J.M. Bakker[3]

[1] Department of Information Systems, 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

[2] Carba-Tec Pty Ltd, 128 Ingleston Rd, Wakerley QLD 4154, Australia.
nrussell@carbatec.com.au

[3] Department of Quality Assurance and Process Innovation, Academic Medical Center, University of Amsterdam, P.O. Box 2260, NL-1100 DD, Amsterdam, The Netherlands. {a.j.moleman,p.j.bakker}@amc.uva.nl

Abstract. In healthcare, processes concerning the diagnosis and treatment of patients can be best characterized as weakly-connected interacting light-weight workflows where tasks reside at different levels of granularity. Moreover, in hospitals many workitems are linked with appointments. To date, Workflow Management Systems (WfMSs) fall short in supporting healthcare processes as no scheduling support and inter-workflow support is offered. To address these problems, we present the YAWL4Healthcare WfMS which supports the seamless integration of unscheduled (flow) and scheduled (schedule) tasks and which allows for dividing complex entangled processes into simple autonomous fragments that may cope with different levels of granularity. Note that our system has been realized by adding significant extensions to the open-source YAWL WfMS.

1 Introduction

In healthcare organizations, many complex are undertaken. Here, Workflow Management Systems (WfMSs) are interesting as, based on process definitions, they are able to manage the flow of work such that individual workitems are done at the right time by the proper person.

However, in order to identify the limitations for applying workflow technology in the healthcare domain we have performed a large case study in which a representative healthcare process of the AMC hospital in Amsterdam, The Netherlands, has been implemented in multiple WfMSs [6]. This exercise revealed two important shortcomings:

First, contemporary WfMSs offer workitems to users via specific worklists. Users select the workitems they will perform without having a schedule in mind. However, in hospitals, many workitems are linked to appointments. Moreover, for these appointments enough time needs to be reserved in which they can be performed in order to prevent the need for rescheduling. Unfortunately, current WfMSs do not provide support for the *calendar-based scheduling of workitems* such that they are performed by *one or more resources* and at a *specified time.*

Second, the process of diagnosing or treating a patient typically consists of the execution of a number of smaller workflow fragments that run in conjunction with each other. Although these fragments execute independently from each other, a certain "magnetic force" exists between them, i.e. these fragments interact with each other. Additionally, these fragments need to cope with different levels of granularity. For example, a doctor may decide during a first visit of a patient that a lab test is needed, the patient needs to be discussed during a multidisciplinary meeting, and that the results of the latter two need to be available for the second visit of the patient. To date, contemporary WfMSs only support monolithic processes and do not offer support for *weakly-connected interacting lightweight workflows* which can cope with *different levels of granularity.*

In order to deal with the two above mentioned shortcomings, we have focused on the general problem of how WfMSs can be extended with facilities for both *scheduling support* and *inter-workflow support* [5, 6]. In order to demonstrate our ideas we have developed a concrete prototype implementation of a WfMS which allows for providing improved support for healthcare processes. This system, called *YAWL4Healthcare*, has been realized by adding significant extensions to the open-source YAWL WfMS [4] and will be the focus of this paper.

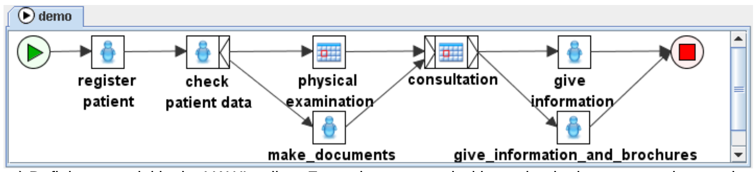
2 The YAWL4Healthcare System

In this section, the YAWL4Healthcare system will be discussed in detail. First, we focus on the provided scheduling and inter-workflow support. Next, we focus on the architecture of the system.

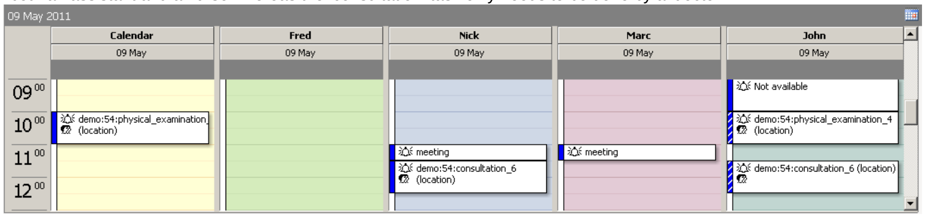
2.1 Scheduling Support

The main scheduling support features will be illustrated using a small scenario for which the corresponding process definition is shown in the YAWL process editor in Figure 1a. As for the "physical examination" and "consultation" tasks a concrete appointment is needed they are annotated with a calendar icon. Moreover, they are called *schedule* tasks. The tasks annotated with a person icon are called *flow* tasks and workitems for them are offered via an ordinary work-list. Moreover, with regard to the scheduling of appointments, for each task its duration is indicated and the "roles" attribute defines the role for each resource that is required to perform the task. For schedule tasks it can also be indicated whether the patient is required to be present.

Assume that an instance of the process has been started and that it has been indicated that the availability of patient "John" needs to be taken into account. As a result, as can be seen in Figure 1, an appointment is booked for the "physical examination" task which appears in the calendars of patient "John", assistant "Jane", and nurse "Sue" and an appointment is booked for the "consultation"



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. Moreover, the 'physical examination' task needs to be done by both an assistant and a nurse whereas the 'consultation' task only needs to be done by a 'doctor'.



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

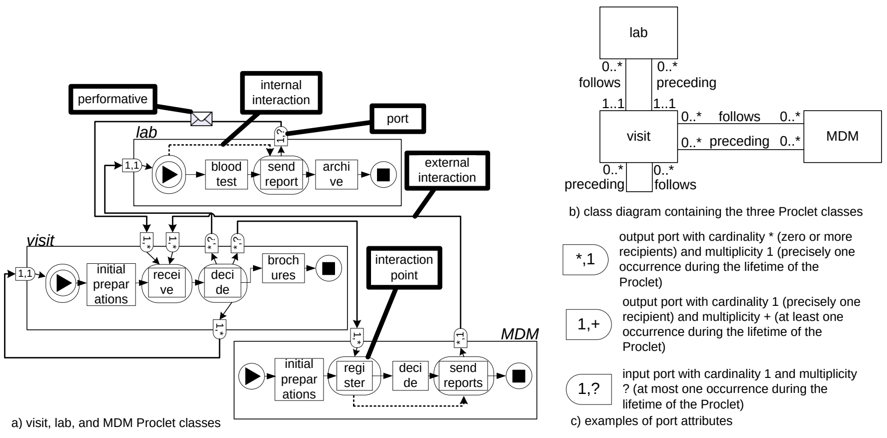
Fig. 1. Scheduling of appointments within the YAWL4Healthcare system.

task which appears in the calendar of patient "John" and doctor "Nick". 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 preceding work-items for a scheduled schedule task, the corresponding appointment is automatically rescheduled. Also, a user can trigger the rescheduling of a task.

2.2 Inter-Workflow Support

Our extensions for augmenting YAWL with inter-workflow support are based on the Proclets framework [3]. First, we briefly introduce the Proclets framework. Proclets provide a framework for modeling and executing lightweight workflows that may reside at different levels of aggregation and interact with each other [3]. A Proclet class specifies which tasks need to be executed and in which order. Here, the process definition of a Proclet class is based on the YAWL language.

Proclet instances interact via *channels*. A channel can be used to send a *performative* (a specific kind of message) to one Proclet instance or a group of Proclet instances. Proclet classes are connected to channels via *ports*. Each outgoing port is connected with exactly one incoming port called an *external interaction*. Furthermore, every port is connected to one *interaction point* representing a specific point in a Proclet class at which interactions with other Proclet classes may take place. Furthermore, for an interaction point having only incoming ports, it may be desired that the receipt of an individual performative is followed by the subsequent sending of a performative at a later point in the execution of that Proclet instance. In that case, two interaction points are connected via an *internal interaction*.



a) visit, lab, and MDM Proclet classes

b) class diagram containing the three Proclet classes

*,1 output port with cardinality * (zero or more recipients) and multiplicity 1 (precisely one occurrence during the lifetime of the Proclet)

1,+ output port with cardinality 1 (precisely one recipient) and multiplicity + (at least one occurrence during the lifetime of the Proclet)

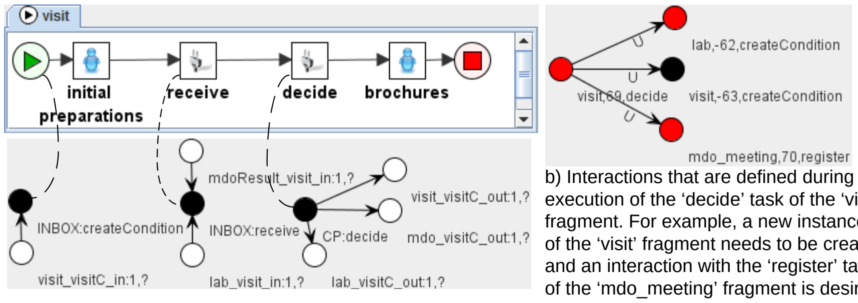
1,? input port with cardinality 1 and multiplicity ? (at most one occurrence during the lifetime of the Proclet)

c) examples of port attributes

Fig. 2. Three Proclet classes: visit, lab, and MDM.

Figure 2 shows three Proclet classes. The "visit" Proclet class describes a visit of a patient to the hospital, the "lab" Proclet class a single lab test that is performed, and the "MDM" Proclet class a multidisciplinary meeting in which multiple patients are discussed.

In our system, the process definition of a Proclet class is defined in the YAWL editor (top of Figure 3a). For tasks and input conditions for which interactions with other Proclet classes are necessary, a so-called interaction point (visualized by a black dot) is defined via the Interaction Definition Editor. Via ports (visualized by a white dot), interaction points are connected with interaction points of other Proclet classes.



b) Interactions that are defined during execution of the 'decide' task of the 'visit' fragment. For example, a new instance of the 'visit' fragment needs to be created and an interaction with the 'register' task of the 'mdo_meeting' fragment is desired.

a) Defining a Proclet model in the YAWL editor and the Interaction Definition Editor. Via dotted arcs, tasks and interaction points are linked with each other. Moreover, interactions points are indicated by black dots whereas ports are indicated by white dots.

Fig. 3. Definition and selection of interactions between workflow fragments.

At run-time, for a certain entity (e.g. a patient or a lab test), interaction points allow for easily nominating interactions with existing and future Proclet instances. That is, for a task instance which is linked with an interaction point which has only outgoing arcs, it is automatically identified which potential interactions with existing and future Proclet instances are possible. Afterwards, the desired interactions can be nominated. For each entity these interactions are saved in an interaction graph (Figure 3b). For example, assume that for patient “Sue” it is decided during the “decide” task of the first visit, that a next visit is necessary, a lab test is required, and that she needs to be discussed during a multidisciplinary meeting. Finally, the result of the lab test and the multidisciplinary meeting need to be available for the second visit. The first part of this scenario is defined in the interaction graph of Figure 3. The “decide” task is represented by the “visit,69,decide” node, the creation of the second instance of the “visit” fragment by the “visit,-63,createCondition” node, the creation of the lab fragment by the “lab,-62,createCondition” node, and the registration for the multidisciplinary meeting by the “mdo_meeting,70,register” node. Note that for the “visit,69,decide” node it is automatically calculated that the aforementioned interactions are possible.

2.3 Architecture

In Figure 4 the architecture of our system is shown. The main components are the following.

- The *Workflow Engine* is the 'core' of the system and takes care of the routing of cases. The engine is realized by using the YAWL WfMS.
- For a task that becomes available for execution, the corresponding work item is communicated to users via the *Workflow Client Application*. This component is realized using Outlook 2003 clients.
- The *Scheduling Service* is responsible for providing scheduling facilities to the WfMS. This service has been realized as a Java service which communicates with the Workflow Engine and the Calendars component via SOAP messages.
- The *Calendar* component is responsible for providing a view on the calendars of users and for manipulating their contents. Here we selected Microsoft Exchange Server 2007 as the system for storing the calendars of users.
- The *Inter-Workflow Service* adds the desired inter-workflow support. Here, the Interaction Service is responsible for managing interactions between Proclet instances at runtime and has been set-up as a YAWL Custom Service. The Interaction Definition Editor offers tools that allow for defining the remaining aspects of Proclet classes on top of process definitions that are defined in the YAWL editor (e.g. interaction points, ports). Additionally, tools are offered such that at instance level human actors can define necessary interactions between Proclet instances. This subcomponent has been implemented as a Java application.

The YAWL4Healthcare WfMS has been developed as a research prototype. So far, we tested the entire system with a number of processes. However, regarding

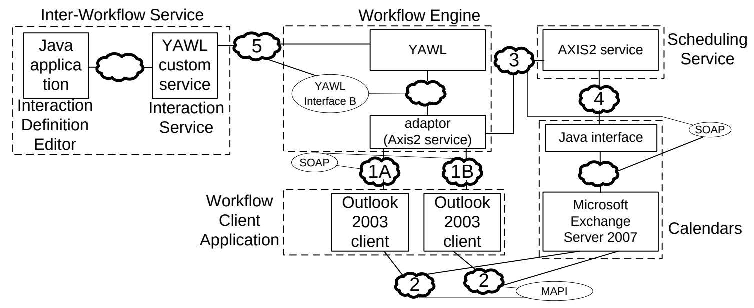


Fig. 4. Architecture of the YAWL4Healthcare system.

the scheduling support related components of the system, an approach has been applied in which the "Workflow Engine" and the "Scheduling service" have been systematically tested [7]. For the future, we plan to evaluate the operation of our resultant system in a real-life scenario at the AMC hospital.

3 Links

For the YAWL4Healthcare system an environment to test and play with the system is provided via SHARE [1]. The environment includes the system itself, a tutorial, two screencasts, and several input models. Furthermore, additional information can be found on [2].

References

1. SHARE Website. http://is.tm.tue.nl/staff/pvgorp/share/.
2. YAWL4Healthcare Website. http://www.processmining.org/yawl4healthcare/start.
3. 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.
4. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N.C. Russell, editors. *Modern Business Process Automation: YAWL and its Support Environment.* Springer-Verlag, 2010.
5. 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.
6. R.S. Mans, W.M.P. van der Aalst, N.C. Russell, P.J. Bakker, A.J. Moleman, K.B. Lassen, and J.B. Jørgensen. From Requirements via Colored Workflow Nets to an Implementation in Several Workflow Systems. *Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) III*, 5800:25–49, 2009.
7. R.S. Mans, W.M.P. van der Aalst, N.C. Russell, P.J.M. Bakker, and A.J. Moleman. Model-based Development and Testing of Process-Aware Information Systems. In *Proceedings of The First International Conference on Advances in System Testing and Validation Lifecycle*, pages 129–134. IEEE Computer Society Press, 2009.