Schedule-Aware Workflow Management Systems

R.S. Mans[1,2], N.C. Russell[1], W.M.P. van der Aalst[1], A.J. Moleman[2], P.J.M. Bakker[2]

[1] Department of Information Systems, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{r.s.mans,n.c.russell,w.m.p.v.d.aalst}@tue.nl
[2] Academic Medical Center, University of Amsterdam, Department of Quality Assurance and Process Innovation, Amsterdam, The Netherlands.
{a.j.moleman,p.j.bakker}@amc.uva.nl

Abstract. Contemporary workflow management systems offer work-items to users through specific work-lists. Users select the work-items they will perform without having a specific schedule in mind. However, in many environments work needs to be scheduled and performed at particular times. For example, in hospitals many work-items are linked to appointments, e.g., a doctor cannot perform surgery without reserving an operating theater and making sure that the patient is present. One of the problems when applying workflow technology in such domains is the lack of calendar-based scheduling support. In this paper, we present an approach that supports the seamless integration of unscheduled (flow) and scheduled (schedule) tasks. Using CPN Tools we have developed a specification and simulation model. Based on this a system has been realized that uses YAWL, Microsoft Exchange Server 2007, Outlook, and a dedicated scheduling service. The approach is illustrated using a real-life case study at the AMC hospital in the Netherlands.

1 Introduction

Healthcare is a prime example of a domain where the effective execution of tasks is often tied to the availability of multiple scarce resources, e.g. doctors. In order to maximize the effectiveness of individual resources and minimize process throughput times, typically an appointment-based approach is utilized for scheduling the tasks performed by these resources. However, the scheduling of these appointments is often undertaken on a manual basis and its effectiveness is critically dependent on preceding tasks being performed on-time in order to prevent the need for rescheduling.

To illustrate the importance of the afore-mentioned issue, consider a small hospital process for diagnosing a patient, shown in Figure 1. As a first step, the patient is registered. Next, a physical examination (task "physical examination") of the patient takes place which is done by an assistant and a nurse. In parallel, a nurse prepares the documents for the patient (task "make documents"). When these tasks have been completed, a doctor evaluates the result of the test (task "consultation") and decides about the information and brochures that need to

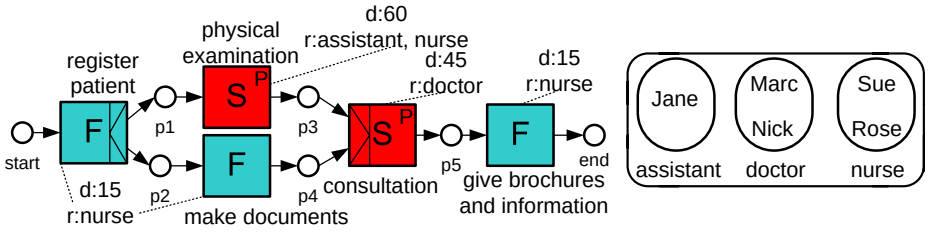


Fig. 1. Running example showing schedule (S) and flow (F) tasks. The prefix "d:" indicates the average time needed for performing the task and prefix "r:" indicates which roles are necessary to perform the task. From each associated role, exactly one person needs to be assigned to the task. For both schedule tasks, indicated by the character "P" in the top-right corner of the task, the patient is also required to be present.

be provided by the nurse (task "give brochures and information"). Figure 1 also shows the corresponding organizational model which specifies the roles being played by people in the organization.

From this example, it can be seen that a distinction can be made between two kinds of tasks. The tasks annotated with an "F" in the figure, can be performed at *an arbitrary point in time when a resource becomes available* and are called *flow tasks*. However, the tasks "physical examination" and "consultation", annotated with an "S" in the figure, can only be performed when the required room is reserved, the patient is present, and the necessary medical staff are present for performing the specific task, i.e. these tasks need to be scheduled and performed at particular times. Therefore, we call these kinds of tasks *schedule tasks* as they are performed *by one or more resources at a specified time*.

For the consultation task in the figure, it is often the case that a doctor finds out at the actual appointment that some results from required diagnostic tests are missing. Consequently, this leads to wasted time for the doctor as a new appointment needs to be scheduled. Therefore, for the effective performance of schedule tasks it is vital that the whole workflow is taken into account in order to guarantee that preceding tasks are performed on-time thereby preventing the need for rescheduling and avoiding unproductive time for resources as a result of canceled appointments.

Workflow technology presents an interesting vehicle with which to support healthcare processes. Based on a corresponding process definition, Workflow Management Systems (WfMSs) support processes by managing the flow of work such that individual work-items are done at the right time by the proper person [2]. Contemporary WfMSs offer work-items through so-called work-lists. At an arbitrary point in time, a user can pick a work-item from this list and perform the associated task.

If we consider the implementation of this process in the context of a WfMS, we find that a significant dichotomy exists in that people are used to working in a scheduled way, but this is not supported by current WfMSs. In contrast

to administrative processes, healthcare processes invoke the coordination of expensive resources which have scarce availability. Therefore, it is of the utmost importance that the scheduling of appointments for these resources is done in an efficient way, that is suitable both for the medical staff and also for the patients being treated. To summarize, there is a *need to integrate workflow management systems with scheduling facilities.*

In this paper, we present the design and implementation of a WfMS supporting both schedule and flow tasks. In addition to the classical work-list functionality generally associated with workflow systems, the concept of a calendar is also introduced in order to present appointments for scheduled work-items to the people involved. Unlike traditional workflow implementations, our focus is on how WfMSs can be *integrated* with scheduling facilities rather than simply extending the functionality of a WfMS or a scheduling system (e.g. a scheduling algorithm). In other words, we investigate how scheduling facilities can be added to workflow systems in general.

An interesting problem in this context lies in the actual development approach taken to extending a WfMS with scheduling facilities. Our strategy for this is based on the use of CPN Tools, a widely used modeling and execution tool for Colored Petri Nets, with which we developed a *comprehensive conceptual model capable of serving both as a specification and simulation model for the application domain.* Formalizing such a system using CP Nets offers several benefits. First of all, building such a net allows for *experimentation.* So, the model or parts of it can be executed, simulated, and analyzed which leads to important insights about the design and implementation of the system. Second, the hierarchical structuring mechanism of CP Nets allows for the modeling of large complex systems at different levels of abstraction. That is, CP Nets can be structured into a set of components which interact with each other through a set of well-defined interfaces, in a similar way to the components in a modular software architecture.

In this way, we were able to use the conceptual model as a *specification* for the subsequent realization of the system. In order to realize the functionality contained in the conceptual model, we *incrementally* mapped it to an operational system based on widely available open-source and commercial-off-the-shelf (COTS) software. Although the conceptual model is detailed, it remains abstract enough, such that its components can be concretized in many different ways. We choose an approach based on the reuse of existing software. In total, the conceptual model consists of 30 nets, 250 transitions, 634 places, and in excess of 1000 lines of ML-code illustrating the overall complexity of the system. For the concrete realization of the system we used the open-source, service-oriented architecture of YAWL and Microsoft Exchange Server 2007 as the implementation platform.

The remainder of the paper is organized as follows. In Section 2 we explain how a workflow language can be augmented with information relevant for scheduling. In Section 3 we present the design of a WfMS integrated with scheduling facilities, together with a concrete implementation. In Section 4 a concrete

application of the realized system is presented. Section 5 discusses related work and finally Section 6 concludes the paper.

2 Flow and Schedule Tasks

In order to allow for the extension of a WfMS with scheduling functionality some concepts need to be introduced. It is assumed that the reader is familiar with basic workflow management concepts, like case, role, and so on [2]. Using the process shown in Figure 1, we will elaborate on how a workflow language can be integrated with scheduling functionality.

2.1 Concepts

We can distinguish between two distinct types of tasks. Flow tasks are performed at an arbitrary point in time when a resource becomes available. As only one resource is needed, it is sufficient to define only *one* role for each of them[3]. Consequently, these tasks can be presented in an ordinary *work-list*. For example, for the flow task "make documents" the work may either be performed by "Sue" or "Rose".

Conversely, schedule tasks are performed by one or more resources at a specified time. As multiple resources can be involved, with different capabilities, it is necessary to specify which kinds of resources are allowed to participate in completing the task. To this end, multiple resources may be defined for a schedule task where for each role specified, only *one* resource may be involved in the actual performance of the task. For example, in Figure 1, the schedule task "physical examination" may be performed by "Jane" and "Rose", but not by "Sue" and "Rose". Note that a resource involved in the performance of a schedule task may also be a physical resource such as medical equipment or a room. Furthermore, for the schedule tasks the patient may also be involved which means that the patient is also a required resource for these tasks. Note that the patient is not involved in the actual execution of the task but is a passive resource who needs to be present whilst it is completed. For this reason, the patient is not added to any of the roles for the task, nor are they defined in terms of a separate role. Instead, it is necessary to identify for which schedule tasks the patient needs to be present.

For presenting the appointments made for schedule tasks to users, the concept of a *calendar* will be used. More specifically, each resource will have its own calendar in which appointments can be booked. Note that each patient also has his / her own calendar. An appointment either refers to a schedule task which needs to be performed for a specific case or to an activity which is not workflow related. So, an appointment appears in the calendars of all resources that are involved in the actual performance of the task. An appointment for a schedule

[3] There also exist approaches for which more roles may be defined, but this is not the focus of our work.

task, for which a work-item does not yet exist, can be booked into the calendar of a resource. However, when the work-item becomes available it has already been determined when it will be performed and by whom. Note that sometimes work-items need to be rescheduled because of anticipated delays in preceding tasks.

In order to be able to determine at runtime the earliest time that a schedule task can be started, information about the duration of every task needs to be known. For example, in Figure 1, for each task the average duration is indicated by prefix "d:". For example, one block represents one minute, which means that the task "physical examination" takes 60 minutes on average.

2.2 Formalization

Based on the informal discussion in the previous section, we now formalize the augmented workflow language. The definition of our language is based on WF-nets [2]. Note that our results are in no way limited to WF-nets and can be applied to more complex notations (BPM, EPCs, BPEL, etc). Note that WF-nets are the most widely used formal representation of workflows. A WF-net is a tuple $N = (P, T, F)$ defined in the following way:

- P is a non-empty finite set of *places*;
- T is a non-empty finite set of *tasks* ($P \cap T = \emptyset$);
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
- There is one initial place $i \in P$ and one final place $o \in P$ such that every place or transition is on a directed path from i to o.

A WF-net can be extended in the following way, called a *scheduling WF-net* (sWF-net). A *sWF*-net is a tuple $N = (P, T_f, T_s, F, CR, Res, Role, R, Rtf, Rts, D)$, where:

- T_f is a finite set of *flow tasks*;
- T_s is a finite set of *schedule tasks*;
- $T_f \cup T_s = T$ and $T_f \cap T_s = \emptyset$, i.e., T_s and T_f partition T. So, a task is either a flow task or a schedule task, but not both;
- (P, T, F) is a WF-net;
- $CR \subseteq T_s$ is the set of schedule tasks for which the human resource for whom the case is being performed is also required to be present.
- Res is a non-empty finite set of *resources*;
- $Role$ is a non-empty finite set of *roles*;
- $R: Res \rightarrow \mathcal{P}(Role)$ is a function which maps resources on to sets of roles;
- $Rtf: T_f \nrightarrow Role$ is a partial function which maps flow tasks on to roles;
- $Rts: T_s \rightarrow \mathcal{P}(Role) \backslash \{\emptyset\}$ is a function which maps schedule tasks on to at least one role;
- $D: T \rightarrow \mathbb{N}_0$ is a function which maps tasks onto the number of blocks that are needed for the execution of the task.

Note that Figure 1 fully defines a particular sWF-net.

3 Design

In this section, we present the design and implementation of a WfMS integrated with scheduling facilities. First of all, the approach followed for doing this is presented. Second, in Section 3.2, we introduce the architecture of the system. Third, for each component identified in Section 3.2, a detailed (functional) description is provided in sections 3.3 to 3.5.

3.1 Approach

Contemporary WfMSs provide a wide range of functions. In order to determine before the implementation phase, how such a system can be integrated with scheduling facilities one needs to identify how the new scheduling functionality being added should be incorporated with existing functionality. To this end, *Colored Petri Nets (CP Nets)* [12] have been chosen as the mechanism to identify and formalize the behavior of the system. CP Nets provide a well-established and well-proven language suitable for describing the behavior of systems exhibiting characteristics such as concurrency, resource sharing, and synchronization.

Formalizing a system using CP Nets offers several benefits. First of all, building such a net allows for *experimentation*. So, the model or parts of it can be executed, simulated and analyzed which leads to insights about the design and implementation of the system. Second, a complete model of the system allows for *testing* parts of the system that are implemented. Given that a CP Net consists of several components, we can "replace" one or more components in the CP Net by the concrete implementation of these components by making connections between the CP Net model and components in the actual system. As the CP Net is an executable model this allows for the testing of numerous scenarios facilitating the discovery of potential flaws in both the architecture and the implementation.

Another important benefit of having a CP Net consisting of several components, is that it provides precise guidance in the configuration of software products, thereby allowing for the use of existing software. As will become clear below, whilst the specification model is detailed, it remains abstract enough, such that it allows components to be concretized in various ways.

3.2 Architecture

In this section, we give a global overview of the architecture of a WfMS integrated with scheduling facilities. The architecture of both the conceptual model of the system and its concrete implementation are shown in Figure 2. Both architectures illustrate the main components and the system is defined in a service oriented way. The components are loosely coupled and the interfaces (shown as clouds) are kept as compact and simple as possible.

In Figure 2b, we see for the actual system implementation how the components have been realized. As the interfaces share the same numbering, it is easy to compare both sets of interfaces.

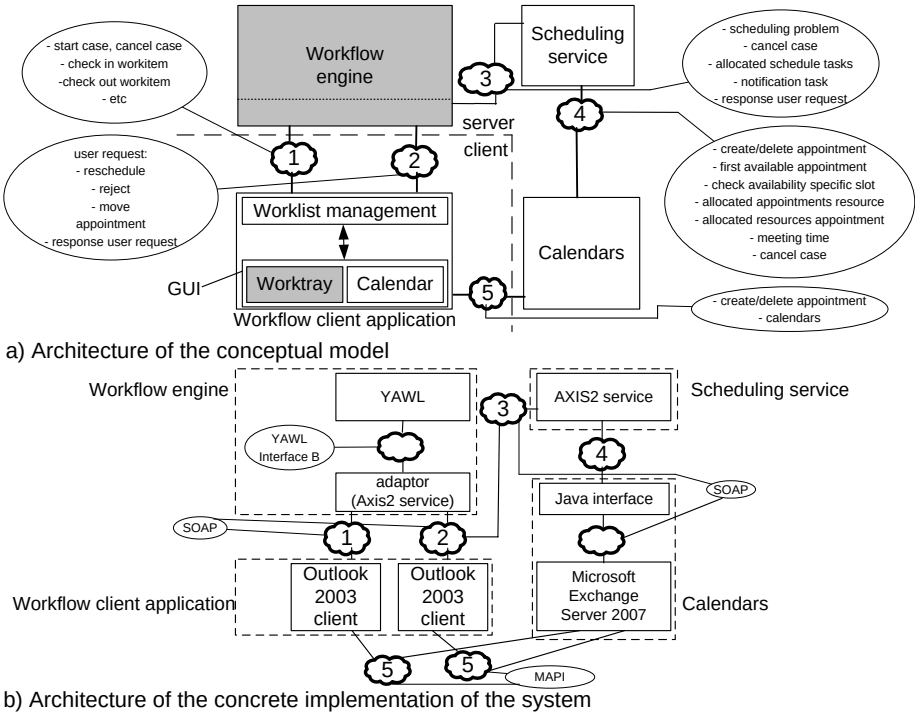


Fig. 2. Architectures of both the conceptual model and the concrete implementation of the system. There are four main components: (I) workflow engine, (II) scheduling service, (III) workflow client application, and (IV) calendars. The distinct interfaces are indicated by numbers.

The architecture consists of four components. First of all, the *workflow engine* routes cases through the organization. Based on the business process definition for a case, tasks are carried out in the right order and by the right people. Once a task in a case becomes available for execution, the corresponding work-item is communicated to users via the *workflow client application* allowing it to be selected and performed by one of them. The scheduling service and the workflow client application communicate with the *Calendar* component in order to obtain a view on users' calendars and to manipulate their contents. Note that users can add /remove appointments that are possibly unrelated to the workflow.

As our focus is on how a WfMS can be *integrated* with scheduling facilities, we want to completely separate the scheduling facilities provided by the system from the engine. As a consequence, we have a separate *scheduling service* component which is responsible for providing scheduling facilities to the system (e.g. (re)scheduling of tasks). In order for the scheduling service to work function correctly, all scheduling constraints imposed by the engine (which might be relevant to a scheduling decision) need to be sent to the scheduling service. To be more precise, the scheduling service receives a *scheduling problem*, which contains all

relevant constraints for one case only. Based on these constraints, the scheduling service makes decisions with regard to the scheduling of schedule tasks for the case.

Informally, the scheduling problem is formulated as a *graph* which has *nodes* and *arcs* between nodes. Nodes, arcs and the graph itself may have properties represented as name-value attributes. The rationale for representing the scheduling problem using this data structure is that any information in the graph can be included which is deemed relevant. For a case, which is in a given state, we map the process definition, defined in terms of the formal definition given in Section 2.2, to the graph (e.g. tasks, duration, split/join semantics of a node, roles). Where a work-item exists for a given node, a property is added to that node indicating the state the work-item is currently in. For a user to reschedule an appointment, additional information is added, such as the name of the requester. Moreover, if the human resource for which the case is being performed is also required in order to perform any task, then the name of the calendar for this resource is included together with the names of the relevant schedule tasks.

An example of a scheduling graph is given in Figure 3. In this figure, we see how the process definition shown in Figure 1 is mapped to the graph. In order to simplify the graph, the figure only shows the properties of the "consultation" node. For this node it indicates that the average duration is 45 minutes, only a doctor is allowed to perform the task, the task is a schedule task, the node has XOR-split semantics, AND-join semantics, and a work-item exists for it which is in the enabled state.

In sections 3.3 to 3.6 the individual components are discussed in more detail. For each component a description of the main functionality is provided together with a discussion on its interaction with other components. Note that, due to space limitations, only the most important interface methods will be discussed.

3.3 Workflow Engine

A workflow engine is responsible for the routing of cases. In addition to the standard facilities an engine should provide [2], the following facilities are added in order to integrate scheduling capabilities.

The engine is responsible for sending a scheduling problem to the scheduling service in order to determine whether appointments need to be (re)scheduled, or if limited time remains in which to finish work-items for preceding tasks

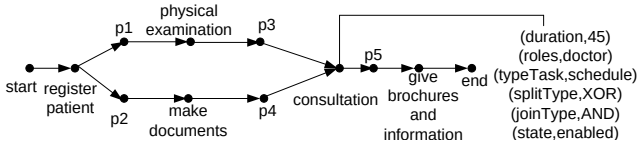


Fig. 3. Scheduling graph for the running example of Figure 1 in which the task "consultation" is enabled.

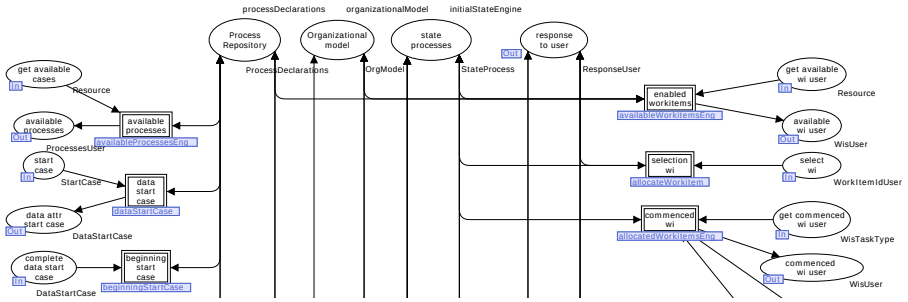


Fig. 4. CP Net component for the workflow engine component.

of an appointment. As a consequence of our choice to completely separate the scheduling facilities from the engine, a scheduling problem for a case is sent when the following situations occur: (1) a case is started; (2) a work-item is finished; (3) a user wants to reschedule an appointment; and (4) at regular time intervals. The fourth option is necessary as it may be the case that no work-items are completed in a given period, but that some appointments need to be rescheduled due to the fact that time has passed. Obviously, the graph is sent the least number of times possible.

As a consequence of the execution of the scheduling service, the engine is informed about appointments for which limited time is left in which to finish work-items of preceding tasks. For these work-items, a warning is sent to the workflow client to indicate that limited time remains in which to finish the work-item.

Model.

A fragment of the CP Net for the workflow engine component is depicted in Figure 4. The places at the far right and far left hand side are part of the interface of the engine with other components. As an indication of the complexity of the engine it is worth mentioning that the flattened substitution transition comprises 54 transitions and 127 places. Moreover, the whole CP Net consists of 217 transitions, 518 places and around 950 lines of ML code. The construction of the whole model required more than three months of work. This underscores the fact that it is a complex system.

Implementation.

The Engine component in the CP Net model can be replaced by a concrete implementation which allows it to be tested. The workflow component is realized (see Figure 2b) using the open-source WfMS YAWL [1] and a service which acts as an adaptor in between YAWL and the workflow client application. The adaptor service communicates with YAWL via "Interface B" [1]. The adaptor also communicates with the scheduling service using SOAP messages. However, the adaptor and the YAWL system are tightly coupled as large volumes of work-item and process related information are exchanged.

3.4 Workflow client application

Users working with the WfMS do so via the workflow client application which delivers the basic facilities that should be provided by this facility [2]. The component consists of a GUI and a work-list management component. The work-list management component serves as a layer between the engine and the GUI and takes care of the communication between them. The GUI component consists of a "worktray" and a "calendar" component where the "worktray" provides the same facilities as a classical worktray. The appointments that are created for schedule tasks are advertised via the calendar. Once a work-item becomes available for such an appointment, it can be performed via the calendar. In our approach, only one user can interact with the WfMS with respect to the completion of the work-item. This prevents concurrency issues where multiple users want to complete the same work-item.

With regard to the appointments that are made for schedule tasks, users can 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. Such a user request can be done as a single action and is the only supported means for the rescheduling of appointments by users. In addition, the workflow client also indicates whether limited time is left in which to undertake work-items in order to meet the schedule. Moreover, users are also allowed to add appointments to the calendar which are not workflow related (e.g. having dinner with friends).

As can be seen in Figure 2a, two interfaces are defined for the communication between the workflow client application and the engine. The interface with number "1" defines the standard communication that takes place between an engine and a workflow client application. The interface with number "2" defines methods added as a consequence of the scheduling facilities developed for the system. For this interface, nothing is stored in the engine when these methods are called.

Model.

The corresponding CP Net model for the work-list management component is fairly complex (and is not shown here): the component's model contains 104 transitions and 225 places.

Implementation.

Similarly, the workflow client application component of the CP Net can be replaced by a concrete implementation. Once the Exchange Server was in place we could easily use the Microsoft Outlook 2003 client to obtain a view of a user's calendar. Furthermore, the Outlook client can be configured in such a way that it can act as a full workflow client application which can communicate with the WfMS via an adaptor service via the exchange of SOAP messages.

3.5 Scheduling Service

The scheduling service is responsible for providing scheduling facilities to the WfMS. Scheduling is done sequentially on a case-by-case base. Once a scheduling

problem is received, the scheduling service needs to determine whether some of the schedule tasks need to be (re)scheduled. Moreover, several distinct issues need to be addressed of which we mention the most important ones.

First of all, the final scheduling of tasks needs to occur in the same order as the sequence of schedule tasks in the accompanying process definition for the case. Moreover, there should be sufficient time between two scheduled tasks. Also, when rescheduling appointments, any preceding constraints need to be satisfied. For example, in Figure 1, it needs to be guaranteed that first the "physical examination" is scheduled, followed by the "consultation" which needs to occur at a later time.

Second, for the actual scheduling of an appointment multiple roles can be specified for a schedule task. For each role specified a resource needs to be selected, i.e., the number of roles determines the number of resources involved in the actual performance of the task. If the patient for which the case is performed also needs to be present at an appointment, then this is also taken into account. The scheduling service only books an appointment in the calendars of these resources who need to be present during the performance of the task (i.e. the performers of the task and the patient (if needed)).

Third, the scheduling service is also responsible for determining whether limited time is left for performing preceding work-items for scheduled tasks. In such a situation, the engine needs to be informed. Moreover, the scheduling service is also informed about the cancelation of a case, so that all appointments related to the case can be removed. When too little time is left for performing preceding work-items for a scheduled schedule task, the corresponding appointment is automatically rescheduled which in this context can be seen to be the most straightforward recovery action. However, different strategies can also be conceived for dealing with such situations. Potential solutions can be found in [15].

In this paper, we focus on integration aspects instead on devising new scheduling algorithms. Nevertheless, to demonstrate the approach that is used for the scheduling of appointments, we will briefly examine the implemented 'naive' scheduling algorithm. Of course it can be envisaged that more advanced scheduling strategies are possible.

The (re)scheduling of appointments is done automatically, which means that there is no user involvement. Starting with the tasks in the graph for which a work-item exists, it is determined which schedule tasks need to be (re)scheduled. Once we know that tasks are able to be scheduled, they are scheduled. Moreover, these tasks are scheduled on a sequential basis in order to avoid conflicts involving shared resources. However, we do not schedule any tasks which occur after a choice in the process as this can lead to unnecessary usage of available slots in the calendar. Moreover, we do not take loops into account.

For the actual scheduling of an appointment, a search is started for the first opportunity where one of the resources of a role can be booked for the respective work-item. If found, an appointment is booked in the calendar of the resource. If the patient for which the case is performed also needs to be present at the

appointment, then this is also taken into account. For example, for Figure 1, if a case is started, an appointment is created for task "physical examination" in the calendars of "Jane", "Sue", and the patient, or "Jane", "Rose" and the patient.
Model.
The CP Net model which models the scheduling service consists of 48 transitions and 144 places. Moreover, modeling the scheduling behavior necessitated writing many lines of ML code, involving around more than 60 hours of work.
Implementation.
The concrete implementation of this component of the CP Net is shown in Figure 2b. Here we see that the component is implemented in Java as a service which communicates with the WfMS via SOAP messages. However, in order to get a view of and to manipulate the calendar, the service also communicates via a Java interface with the Exchange Server which in turn exchanges information via SOAP messages.

3.6 Calendar

The Calendar component is responsible for providing a view on the calendars of users and for manipulating their contents. It is possible to create / delete appointments or to get information about the appointments that have been made. Moreover, the interface contains some convenience methods for deleting cases and finding the first available slot for a schedule task. Otherwise, large volumes of low-level information need to be exchanged whereas now only one call is necessary.
Model.
The CP Net model which models the scheduling service consists of 11 transitions and 22 places. Note that this model is relatively simple.
Implementation.
For the Calendar component we selected Microsoft Exchange Server 2007 as the system for storing the calendars of users. The big advantages of this system are its widespread use and the fact that it offers several interfaces for viewing and manipulating calendars.

4 Application

In this section, we demonstrate our approach and software in the context of a real-life healthcare scenario. To evaluate our approach, we have taken the diagnostic process of patients visiting the gynecological oncology outpatient clinic at the AMC hospital, a large academic hospital in the Netherlands. This healthcare process deals with the diagnostic process that is followed by a patient who is referred to the AMC hospital for treatment, up to the point where the patient is diagnosed, and consists of around 325 activities. However, for our scenario we will only focus on the initial stages of the process shown in Figure 5.

At the beginning of the process, a doctor in a referring hospital calls a nurse or doctor at the AMC hospital resulting in an appointment being made for

the first visit of the patient. Several administrative tasks need to be requested before the first visit of the patient (e.g. task "first consultation doctor"). At the first consultation, the doctor decides which diagnostic tests are necessary (MRI, CT or pre-assessment) before the next visit of the patient (task "consultation doctor"). Note that for the MRI, CT and pre-assessment tasks we do not show the preceding tasks at the respective departments that need to performed in order to simplify the model presented.

For this scenario, we assume that the task "additional information and brochures" has been performed. Moreover, at the first consultation with the doctor it has been decided that an MRI and a pre-assessment are needed for the patient. So, by looking at the process model it becomes clear that the tasks "MRI", "pre-assessment" and "consultation doctor" need to be scheduled. The result of the scheduling performed by the system for these tasks is shown in Figure 6a. Note that our case has "Oncology" as its process identifier and has "15" as its case identifier. Moreover, for the "consultation doctor", "pre assessment", and "MRI" examination, a doctor, an anaesthetist, and MRI machine are needed respectively. Moreover, the patient is also required to be present.

In Figure 6a we can see that the "MRI" has been scheduled for 10:00 to 10:45 (see first column), the consultation with the doctor has been scheduled for 13:00 to 13:30 in the calendar of doctor "Nick" (see second column), and that the pre-assessment has been scheduled for 11:00 to 11:30 in the calendar of anaesthetist "Jules" (see third column). At the far right, we can see the calendar of patient Anne who also needs to be present for the work-items mentioned, which explains why the previously mentioned appointments are also present in her calendar. For Anne we see that she is not available till 10 'o clock which has influenced the actual scheduling. This is due to the fact that she can not manage to be at the hospital before 10 'o clock by public transport. However, it is important that the "consultation doctor" task is scheduled after the "MRI" and "pre-assessment" task, which is also consistent with the corresponding process definition.

Now, let us assume that unexpectedly some maintenance for the MRI machine is necessary for that day, which will take until 13:30 hours to complete.

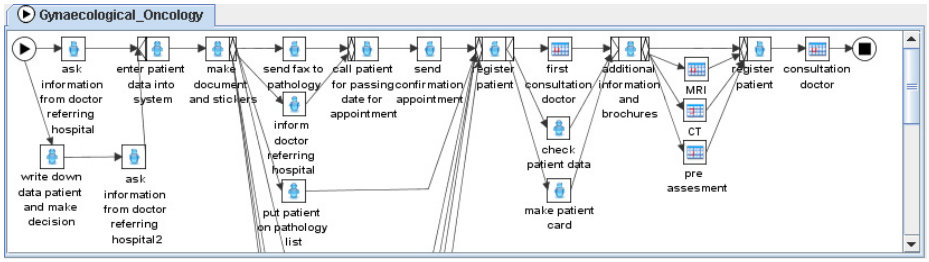
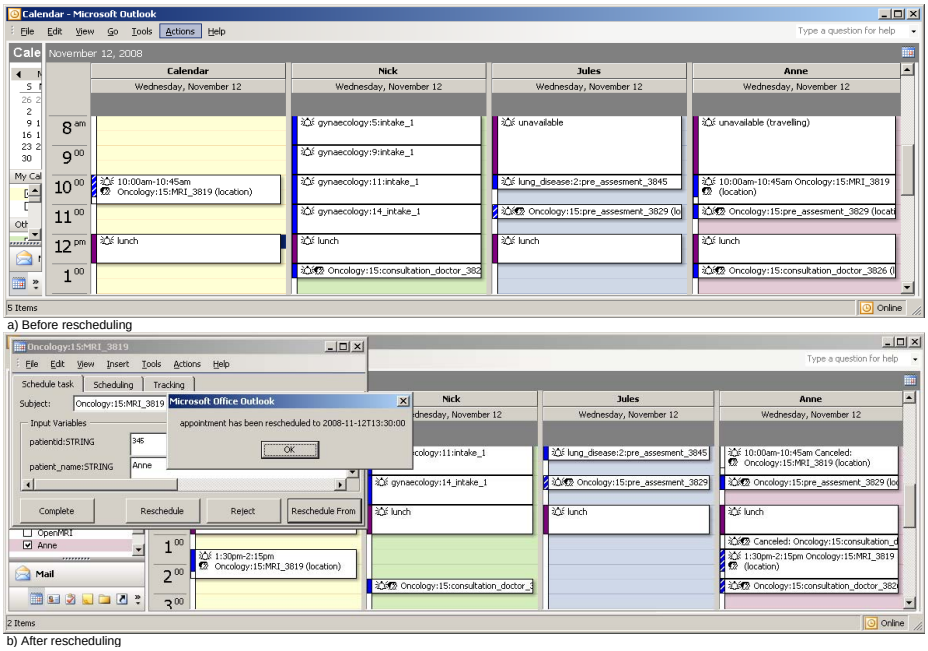


Fig. 5. Screenshot of the YAWL editor showing the initial stages of the gynaecological oncology healthcare process. The flow tasks are indicated by a person icon and the schedule tasks are indicated by a calendar icon.

.



a) Before rescheduling

b) After rescheduling

Fig. 6. Screenshot of the calendars for the MRI, consultation with the doctor, and the pre-assessment before and after rescheduling.

.

Consequently, the MRI appointment needs to be rescheduled to 13:30 hours. The effect of this specific rescheduling request can be seen in Figure 6b. In this figure, the message box indicates that the MRI has been successfully rescheduled to the requested time. Moreover, in the calendar of Anne we can see that the MRI now takes place from 13:30 to 14:15. However, it was also necessary to reschedule the appointment with doctor "Nick" which will now take place from 14:30 to 15:00. As can be seen in Figure 5, this rescheduling step is necessary as the task "consultation doctor" occurs after the "MRI" task and the task "register patient" falls in between these two tasks and takes 15 minutes.

5 Related Work

Analysis of the healthcare research shows that significant work has been done on the problem of appointment scheduling. Examples of such research efforts are appointment scheduling for outpatient services [7] and operating room scheduling [6]. However, most of these studies focus on a single unit instead of situations in which a patient may pass through multiple facilities. In our research, we take the scheduling of work-items for the whole workflow into account together with the current state of a case.

Our work is also related to time management in workflows. For example, in [13, 11] the authors focus on the satisfiability of time constraints and the enforcement of these at run-time. In addition, there is also research on the problem of the scheduling of tasks by WfMSs. For example, [4, 9, 14] present algorithms for the scheduling of tasks. In contrast, we focus on the augmentation of a WfMS with scheduling facilities instead of just presenting new scheduling algorithms.

The work presented in [8] is somewhat similar to ours as it presents different architectures for a WfMS in which temporal aspects are explicitly considered. However, the temporal reasoning facilities are added as core functionality to the engine. In this paper, we propose a different approach where this kind of functionality is realized through a separate service in the system. In this way, loose coupling is guaranteed which means that our approach can be generalized to any WfMS (or even to multiple engines at the same time).

Multiple people can be involved in the actual performance of a schedule task. However, in our approach, only one user can interact with the WfMS with respect to the completion of a work-item. In [3, 5, 10] reference models to extend the organizational meta model with a team concept allowing for the distribution of work to teams are proposed. By doing so, advanced mechanisms are offered for the performance of work by such a team. Additionally, in [3, 5] a language is discussed for defining work allocation requirements to people.

6 Conclusions and Future Work

In this paper, we have presented the design and implementation of a WfMS augmented with calendar-based scheduling facilities. Instead of just offering work-items via a work-list, as is the case in most existing WfMSs, they can also be offered as a concrete appointment in a calendar taking into account which preceding tasks are necessary and whether they have been performed.

Our approach demonstrates that the use of CP Nets, for constructing a conceptual model of the system to be realized, provides valuable insights in terms of understanding the problem domain and identifying the behavior of the system. Moreover, the same conceptual model provides a comprehensive specification on which to base the ultimate realization of the required functionality. We have incrementally mapped it to an operational system using widely available open-source and commercial-off-the-shelf (COTS) software. This demonstrates that although the specification model is detailed, it remains at a sufficient level of abstraction to allow its constituent components to be concretized in various ways. Moreover, it also shows that our ideas can, for example, be applied to a variety of WfMSs and scheduling systems.

The resultant system has been tested using several realistic scenarios. We plan to test the components of the system in a more systematic way by incrementally "replacing" components of the CP Net by their concrete implementation. In this way, we can test numerous scenarios facilitating the discovery of flaws both in individual components as well as in the overall architecture of the actual system.

In the design and the implementation of the system, a naive algorithm has been used for the scheduling of appointments. This naive approach can lead to inefficient use of resources. In the future, we plan to use the CP Net for evaluating various scheduling approaches and to investigate the effects of our calendar-based approach on case performance.

Finally, to test the feasibility of our approach, we plan to evaluate the operation of our resultant system in a real-life scenario at the AMC hospital.

References

1. W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and Implementation of the YAWL System. In *Proceedings of CAiSE'04*, 2004.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, 2002.
3. W.M.P. van der Aalst and A. Kumar. A Reference Model for Team-Enabled Workflow Management Systems. *Data and Knowledge Engineering*, 38(3):335–363, 2001.
4. C. Bettini, X.S. Wang, and S. Jajodia. Temporal Reasoning in Workflow Systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002.
5. J. Cao, S. Zhang, and X. Zhang. Team Work Oriented Flexible Workflow Management System. In X. Meng, J. Su, and Y. Wang, editors, *Advances in Web-Age Information Management*, volume 2419 of *LNCS*, pages 189–200, 2002.
6. B. Cardoen, E. Demeulemeester, and J. Beliën. Operating Room Planning and Scheduling: A Literature Review. FEB Research Report KBI 0807, Katholieke Universiteit Leuven, Leuven, 2008.
7. T. Cayirli and E. Veral. Outpatient Scheduling in Health Care: A Review of Literature. *Product Operations Management*, 12(4):519–549, 2003.
8. C. Combi and G. Pozzi. Architectures for a Temporal Workflow Management System. In H. Haddad, A. Omicini, R.L. Wainwright, and L.M. Liebrock, editors, *Proc. of the 2004 ACM symposium on applied computing*, pages 659–666, 2004.
9. C. Combi and G. Pozzi. Task Scheduling for a Temporal Workflow Management System. In *Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 61–68, 2006.
10. L. Cui and H. Wang. Research on Cooperative Workflow Management Systems. In W. Shen, Z. Lin, and J.-P.A. Barthès, editors, *Computer Supported Cooperative Work in Design I*, volume 3168 of *LNCS*, pages 359–367, 2005.
11. J. Eder, E. Panagos, and M. Rabinovich. Time Constraints in Workflow Systems. In M. Jarke and A. Oberweis, editors, *Proceedings of CAiSE '99*, volume 1626 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag, Berlin, 1999.
12. 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.
13. O. Marjanovic and M. Orlowska. On Modeling and Verification of Temporal Constraints in Production Workflows. *Knowledge and Information Systems*, 1(2):157–192, 1999.
14. P. Senkul and I.H. Toroslu. An Architecture for Workflow Scheduling under Resource Allocation constraints. *Information Systems*, 30(5):399–422, 2005.
15. W.M.P. van der Aalst, M. Rosemann, and M. Dumas. Deadline-based Escalation in Process-Aware Information Systems. *Decision Support Systems*, 43(2):492–511, 2007.