Augmenting a Workflow Management System with Planning Facilities using Colored Petri Nets

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. {p.j.bakker,a.j.moleman}@amc.uva.nl

Abstract. Traditional workflow management systems distribute workitems to users via a worklist and leave the actual timing of workitem execution to the individual resource(s) performing the task. In work environments in which resources are scarce, expensive and multiple resources are necessary to undertake the workitem, often an appointment-based approach is utilized in order to maximize resource utilization. To this end, we propose the extension of a workflow management system with planning and monitoring facilities in order to guarantee effective resource utilization and minimize dead-time for resources as a result of canceled appointments. This paper describes the approach taken in which first a conceptual model for these extensions has been developed which is based on Colored Petri Nets. Second, based on the conceptual model, a prototype has been developed using YAWL and the collaborative software product Microsoft Exchange Server 2007. The applicability of the approach for the development of large scale systems will be demonstrated by elaborating on the conceptual model and the experiences that have been gained. Finally, the operation of the system is demonstrated in the context of a real-life healthcare scenario.

1 Introduction

Nowadays, hospitals are focusing on improving the quality of care and the service that is delivered to a patient. Typically, when making appointments for a patient, patient preferences are taken into account. Examples of this include considering whether appointments can all be scheduled on one day, and querying the availability of the patient. Similarly, constraints imposed by doctors, nurses, rooms, and medical equipment also need to be considered.

Usually, the scheduling of these appointments is done on a manual basis and does not take into account which preceding tasks are necessary and whether they have been performed. For example, in order to perform surgery it is important that the patient is first seen by an anaesthetist in order to determine the anaesthetics that are needed. Moreover, prior to surgery a last check is performed before proceeding with the final preparations for the operation. If these tasks are not performed in time, this can lead to a delay in performing the operation. Another example occurs where during a regular meeting to discuss the status of patients and plan subsequent treatment, a doctor finds out that not all required information is available. The above mentioned examples lead to the inefficient use of scarce, expensive resources, and necessitate the rescheduling of appointments.

Workflow management systems support process execution by managing the flow of work such that individual workitems are done at the right time by the proper person [1]. The benefits being that processes can be executed faster, more efficiently, and their progress can be monitored. Based on business process definitions, which define the ordering between the

tasks which need to be performed, so called *workitems* are distributed to resources (typically people) for execution. A workitem is an indivisible task of work and corresponds to a task which needs to be performed in the context of a given case. An example of a workitem is the performance of a "CT-scan" for patient "Rose". Typically, the user sees available workitems via a so called *worklist*, which can be seen as a to-do list in which people can view the various workitems that they need to perform. At an arbitrary point in time, a user can pick a workitem and perform the task associated with it.

In healthcare the actual execution of a workitem is often linked to an appointment in which several people can be involved. In other words, an appointment-based approach is often utilized for scheduling workitem execution due to the scarce and limited availability of resources that are involved. However, this schedule-based way of working is not supported by the worklist approach offered by current workflow management systems. Moreover, patient preferences need to be taken into account when making these appointments. Consequently, we need to *extend workflow management systems with planning facilities*. Furthermore, planned appointments need to be monitored to ensure that preceding tasks, in the corresponding process definition, are performed on time. If limited time is left to complete them, this needs to be signalled. If they can not be performed on-time, the appointment and possibly subsequent appointments will need to be rescheduled, which is highly undesirable. So, in addition to planning facilities there is the need to incorporate monitoring facilities as well. Note that the focus is on how workflow management can be *integrated* with scheduling and monitoring facilities instead of extending the functionalities of a workflow management system or a planning system.

In this paper, we present the approach taken to *design* and *implement* a workflow system offering (re)scheduling and monitoring facilities. Moreover, the appointments made can also be shown to the people involved. Figure 1 sketches the approach that has been used.

First of all, we started by augmenting a workflow language with planning functionality. Then, we created a conceptual model of a workflow management system augmented with planning and monitoring facilities. The conceptual model is based on Colored Petri Nets (CPNs) [9] thus providing a complete and formal specification of the system to be implemented. The complete specification of the system in CPNs consists of 27 pages, 377 transitions, 169 places, and over 1000 lines of ML code. The construction of the whole model was undertaken by one person, with advanced knowledge about CPNs and CPN Tools, in about 3 months. This, together with the size of the model, illustrates the overall complexity of the system. Finally, we build a prototype of the system. For this prototype we used the open-source, service-oriented architecture of YAWL [2] and the Microsoft Exchange Server 2007 together with several Outlook 2003 clients. The implementation of the system was done by one person, having already built several software tools, in around three months.

The paper proceeds as follows: Section 2 introduces the research approach that was followed. Section 3 describes how a workflow language can be augmented with planning functionality, followed by a description of the conceptual model, constructed in CPNs, in Section

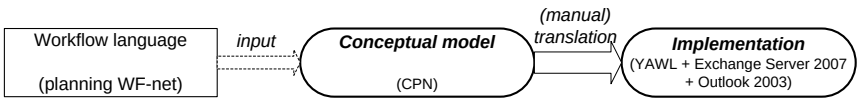

Fig. 1. Overall approach. The workflow language serves as input for the conceptual model. The conceptual model is used as design model for the implementation of the system.

4. Section 5 elaborates on the implementation of the system and outlines a concrete application of the realized system. Section 6 presents related work. Finally, Section 7 discusses the experiences of following the aforementioned approach and concludes the paper.

2 Approach

In this section, we elaborate on the approach that has been followed, as shown in Figure 1, to provide a concrete implementation of a workflow management system augmented with planning and monitoring facilities. As can be seen in the figure, a model-based approach has been used, in which intermediate models are used in order to obtain the final implementation.

The first step was to get insight into how a workflow language could be augmented with planning functionality. As Petri nets are extensively used in Workflow (WF) modeling, primarily because of their mathematical definition and graphical representation, WF-nets were chosen as the basis for these extensions. The main advantage of this choice was that it assisted in the formalization of the augmented workflow language. Note that in principle these extensions can be applied to any workflow language.

Next, we constructed the conceptual model of the system to be realized. A conceptual model serves in understanding a problem domain and identifying how functionality can be added which should collaborate with already existing functionality. The conceptual model that has been constructed is based on CPNs [9]. CPNs provide a well-established and well-proven language suitable for describing the behavior of systems involving characteristics such as concurrency, resource sharing, and synchronization. In this way, they are an excellent candidate for the formalization of such a system.

The CPN language is supported by the CPN Tools offering [9] which we used for creating, simulating and analyzing the model being constructed. This setting allows for *experimentation*, during which deep insights and a good understanding of the design and behavior of the system can be gained. Additionally, it allows for *rapid prototyping*. A complete model of the system can be executed, simulated and analyzed. Flaws in the design can be detected and fixed, leading to a more complete specification. Finally, the system has been implemented using YAWL, Microsoft Exchange Server 2007 and several Outlook 2003 clients by (manually) translating the conceptual model into a working system.

It is important to note that the workflow language is at a different level of abstraction to the conceptual model and the implementation. The workflow language is used as input for the conceptual model. However, the conceptual model and the implementation operate at a similar level. The conceptual model is in such a level of detail that it completely specifies the behavior of the system to be implemented. So, on the basis of the conceptual model, we immediately implement the desired functionality and no other graphical models have been used other than the conceptual model and the implementation. The difference between them is that the conceptual model abstracts from implementation details and language specific issues. The advantage of this is that for the conceptual model we only need to consider the functionality that will be provided by the system and that for the implementation we only need to focus on realizing a working system.

3 Workflow Language

In order to extend workflow systems with planning functionality some new terminology and concepts need to be introduced. We will use a running example for this purpose. It is assumed

that the reader is familiar with basic workflow management concepts, like “case”, “role”, and so on [1].

3.1 Flow and Schedule tasks

Figure 2 outlines a hospital process in which a patient suspected to be suffering from a lung disease is diagnosed.

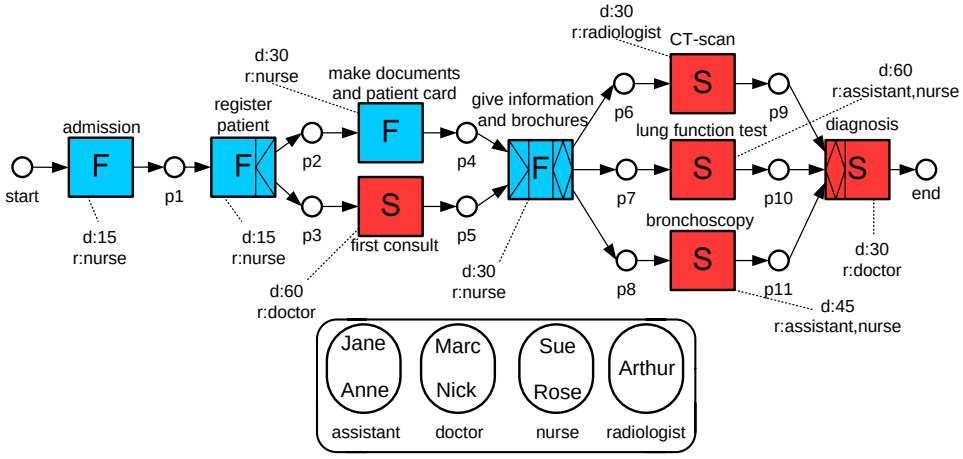

Fig. 2. WF-net for the running example showing schedule (S) and flow (F) tasks. The prefix “d:” indicates the average time needed for performing the task and prefix “r:” indicates which roles are necessary to perform the task. From each associated role, exactly one person needs to be assigned to the task. Note that the “register patient” and “give information and brochures” tasks have XOR split and join semantics associated with them. Moreover, the “give information and brochures” and “diagnosis” tasks have OR split and join semantics. Furthermore, for all of the schedule tasks, the patient is required to be present.

As can be seen in the figure, first the admission is done by a nurse, i.e., some patient related data is recorded and an appointment is made for the first visit of the patient (task “admission”). The next step is that the patient arrives at the outpatient clinic for the first appointment with the doctor (task “register patient”), followed by the first appointment with the doctor (task “first consult”). In this step, a decision is made about the tests to be performed before the second visit of the patient. In parallel, a nurse prepares the documents and the patient card (task “make documents and patient card”). Afterwards, a nurse provides information and brochures to the patient (task “give information and brochures”). Next, the diagnostic tests are performed which are needed for the diagnosis of the patient which is performed by a doctor (task “diagnosis”). For these diagnostic tests a choice can be made from the following tests: CT-scan (task “CT-scan”), lung function test (task “lung function test”), or bronchoscopy (task “bronchoscopy”).

From this example, it can be seen that two kinds of tasks can be distinguished: *flow* tasks and *schedule* tasks. In the figure, a flow task is labeled by an “F” and a schedule task is labeled by an “S”.

Tasks with an "F" annotation should be performed as soon as a resource is able to undertake them. For example, task "make documents and patient card" can be performed by any nurse when task "register patient" is finished. Basically, a flow task can be performed at *an arbitrary point in time when a resource becomes available and there is no reason to postpone it to a specific point in time*. These tasks can be presented in an ordinary worklist where a given resource can start working on the task when it becomes available. Therefore, as only a single resource is needed to perform the task, it is sufficient to define only one role for them. For example, for the flow task "make documents and patient card" only a single nurse is needed which explains why the "nurse" role has been defined.

The tasks annotated by an "S" in the figure correspond to schedule tasks. For these tasks typically multiple resources are required, with different capabilities. A schedule task needs to be performed by *one or more resources at a specified time*. As multiple resources can be involved in the actual performance of a schedule task, at least one role needs to be defined for each of them. For each role specified, only *one* resource may be involved in the actual performance of the task. For example, for task "lung function test" an appointment is needed in which one assistant and one nurse are involved which explains why the "assistant" and "nurse" roles are defined. Note that for the schedule tasks the patient may also be involved which means that the patient is also a required resource for these tasks. The patient is not involved in the actual execution of a task but is only a passive resource who needs to be present. For that 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 which schedule tasks the patient needs to be present for.

Flow tasks are presented in an ordinary worklist. However, schedule tasks are presented in a calendar as for each of them specific appointments need to be made involving multiple resources. Each resource has its own specific calendar in which the appointments made for schedule tasks can be seen. In this way, a single appointment made for a schedule task can appear in multiple calendars but only in the calendars of the resources which are involved in the actual performance of the task even though a workitem does not yet exist. When the workitem becomes available, the schedule task can be performed. Note that an appointment in a calendar may also refer to an activity which is not workflow related.

For the booking of appointments in a calendar, it is important to mention that a calendar consists of *blocks* of equal length. So, all blocks represent the same timeperiod. So, a block may either represent one hour but also one minute. Depending on the length of an appointment and the timeperiod of a block, an appointment may occupy several blocks. For example, the "first consult" task has a duration of 60 blocks if a block represents one minute.

To be more precise, for the correct scheduling of appointments for schedule tasks it must be known at runtime what the estimated duration is of the appointment and what the earliest time is that the appointment may be booked. Therefore, for every task in the process model an average duration needs to be specified. As we use the notion of blocks in calendars, we specify the duration of tasks in terms of blocks. In Figure 2 for each task this is indicated by prefix "d:". For example, one blocks takes 1 minute and task "make documents and patient card" requires 30 blocks which means that the task takes 30 minutes on average to complete.

For reasons of simplicity we only include the average task duration for a task to complete. Ideally, more information on the probability distribution could be used, e.g., the standard deviation.

3.2 Formalization

In this section, a formalization of the augmented workflow language will be presented. A WF-net is a Petri net with one initial and one final place such that every place or transition is on a directed path from the initial to the final place [1]. The execution of a case is represented as a firing sequence that starts in the initial marking, consisting of a single token in the initial place. The token in the final place with no tokens left in the other places indicates proper termination of case execution. A model is called sound if every reachable marking can terminate properly.

WF-net extended with the schedule and flow tasks is called a *planning WF-net* (pWF-net). A pWF-net is a tuple $N = (P, T_f, T_s, F, CR, Res, Role, R, Rtf, Rts, D)$, where

- P is a non-empty finite set of *places*;
- 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 $T_f \cap T_s = \emptyset$ and $T_f \cup T_s \neq \emptyset$, i.e., T_s and T_f partition T. So, a task is either a flow task or a schedule task, but not both. Moreover, the set T may not be empty;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
- $CR \subseteq T_s$ is a set of schedule tasks for which the human resource for whom the case is being performed is also required to be present. For our healthcare example this means the schedule tasks for which the patient is required to be present during their execution.
- 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 onto sets of roles;
- $Rtf: T_f \nrightarrow Role$ is a partial function which maps flow tasks onto roles;
- $Rts: T_s \rightarrow \mathcal{P}(Role) \backslash \{\emptyset\}$ is a function which maps schedule tasks onto at least one role;
- $D: T \rightarrow \mathbb{N}$ is a function which maps tasks onto a the number of blocks that are needed for the execution of the task. This value indicates the average time it takes to execute the task. One block corresponds to a specific actual duration, e.g. a block can be half an hour or one minute[3].

The running example in Figure 2 can be easily mapped onto this formalization. For example, the "lung function test" task belongs to T_s, and where $Rts(lung\_function\_test) = \{assistant, nurse\}$ and $D(lung\_function\_test) = 60$.

4 Conceptual Model in Colored Petri Nets

The conceptual model which defines the precise behavior of a workflow management system augmented with planning facilities is defined in terms of a CPN model. The complete CPN model consists of a series of CPNs in which several layers can be distinguished. Figure 3 shows the hierarchy of CPNs in the CPN model, together with the relationships between them. In total, there are 27 distinct CPNs. An indication of the complexity of each net is expressed by the p and t value included for each them, showing the number of places and transitions they contain. It is not possible to discuss all the nets in details in this paper. Only the blocks in Figure 3 which are colored grey will be discussed. However, this is sufficient to give an overview of the operation of the model. At the end of the section, Section 4.4, we will focus on the analysis of the conceptual model.

[3] Currently, we only use the average value for (re)planning. However, in the future we plan to utilize more information (variance etc).

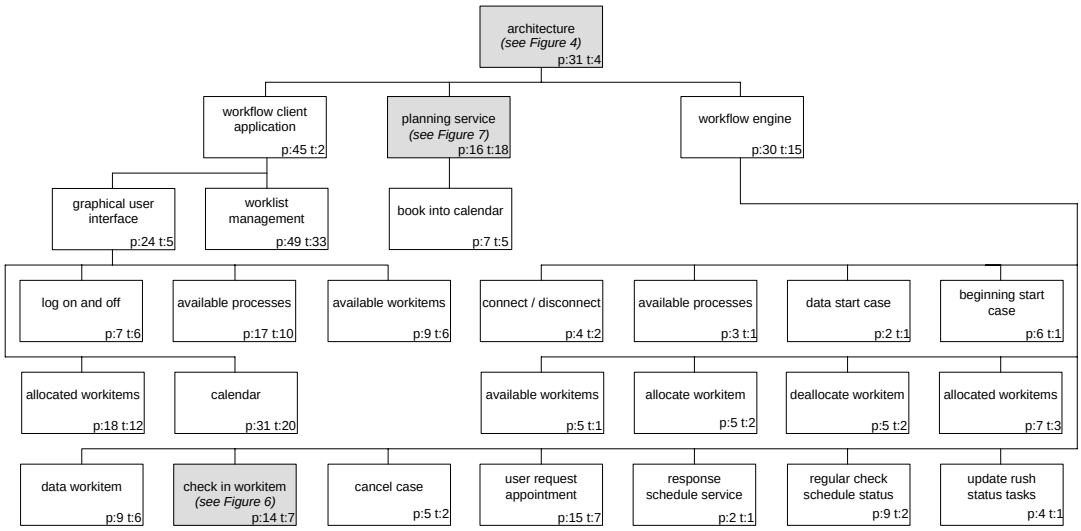

Fig. 3. CPN hierarchy of the conceptual model: each square represents a (sub)net containing places and transitions.

4.1 Overview

Figure 4 shows the topmost net in the CPN model and gives an idea of the main components in the system and the interfaces between them. It can be seen in the figure that there are three substitution transitions. They represent the major functional units in the system, namely: workflow engine, workflow client application and planning service. Each place which is connecting two components forms part of the interface between the two components, except for the place "calendars users" which stores the calendars for each user. The components of the system are set-up in a service-oriented way such that the workflow client application and planning service can interchange data with the workflow engine on a loosely coupled basis. In order to guarantee this, the interface, which defines how two components should interact, should be as minimal as possible. However, this has the advantage that the components can easily be coupled with any other workflow engine component.

The conceptual model consists of three main components.

- The **workflow engine** is the most important component of the workflow system as it is the heart of the system. Based on the business process definition, the engine routes cases through the organization and ensures that the tasks of which they are comprised are carried out in the right order and by the right people. Next to this, the engine takes care of offering workitems to users, once they become available for execution.
- The **workflow client application** communicates the distributed workitems to the users so that they can select and perform them. In our case, workitems that correspond to flow tasks are advertised via the worktray. The appointments that are created for schedule tasks are advertised via a calendar. Once a workitem becomes available for such an appointment, the work can be performed. However, where appointments have been made, 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 data and time, or (3) the reassignment of the appointment to another employee.

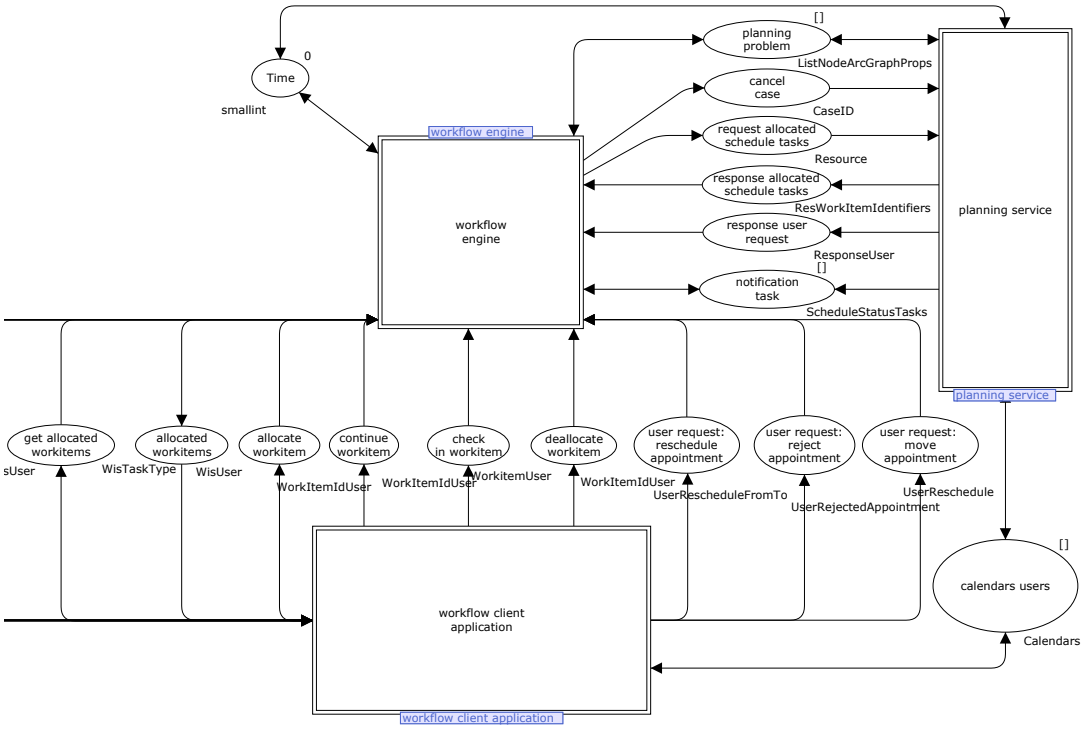

Fig. 4. Overview of the conceptual model.

In addition, the workflow client also indicates whether limited time is left in which to undertake workitems related to preceding tasks for an appointment.
The users who utilize the workflow system interact with it via the workflow client application. All allowed user actions are modeled in subnets of the "graphical user interface" net, which in its turn is a subnet of the "workflow client application" net (see Figure 3).

– The **planning service** component provides the planning capabilities needed by the system. The planning service behaves in a passive way and its operation must be explicitly triggered. Its operation is initiated by the engine which sends a planning problem for a specific case. This planning problem is represented as a graph indicating the planning constraints which hold between the tasks in the corresponding process definition for the case, e.g. the ordering between tasks. Based on this graph, the planning service is responsible for determining whether appointments need to be (re)scheduled. Moreover, the planning service identifies whether limited time is left for the completion of workitems for preceding tasks of an appointment. For such workitems, a warning is forwarded to the users via the workflow engine.

An example of a planning problem that is sent from the workflow engine to the planning service can be found in Figure 5. As can be seen in the figure, the planning graph is formulated as a graph having nodes and directed arcs between the nodes. Additionally, the graph, the nodes and the arcs may have properties. These properties are represented as name-value attributes. In this way, we can add additional constraints to the graph which are relevant for the planning activity. For correct planning of a case, the ordering of tasks in a given process

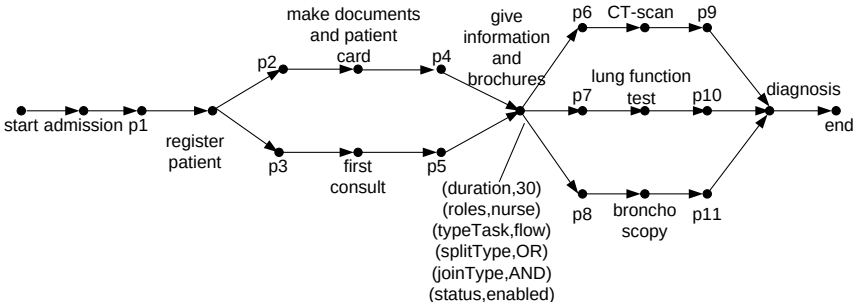

Fig. 5. Planning graph for the running example in Figure 2. The "give information and brochures" task is currently enabled.

definition is relevant. Therefore, for the corresponding process definition of a case, we map all nodes and arcs of the process definition to the graph. If the human resource for which the case is being performed is also required in order to perform some of the schedule tasks, then the name of the calendar for this resource is included together with the names of the relevant schedule tasks. If a workitem exists for a certain node, this is also included in the graph as only this task or subsequent tasks need to be (re)scheduled. Additionally, the following information for a task is included: split and join semantics, whether the node represents a schedule, flow, or dummy (i.e. routing) task, and the roles which are involved in performing the task. So, in Figure 5, we can see how the graph of Figure 2 is mapped to a planning graph. For the "give information and brochures" task it is shown that the average duration is 30 minutes, only a single nurse is needed to execute the task, it is a flow task exhibiting OR split and AND join semantics, and a workitem exists which is in the enabled state. In order to simplify the graph, the properties of the other nodes have not been shown.

4.2 Workflow Engine

Figure 3 shows that the workflow engine comprises of 15 distinct CPNs. In total, they consist of 125 places and 54 transitions thus illustrating that the engine demonstrates fairly complex behavior. The naming of the different subnets in the workflow engine CPN gives a good overview of the functionality that is provided by the workflow engine as they all appear as substitution transition on the workflow engine subpage. It is not possible to describe all aspects of these subnets in detail. Hence, we focus on a specific subnet, the checking in of workitems (substitution transition "check in workitem"), which will be discussed in detail.

Before discussing the operation of this subnet, it is important to mention that a workitem passes through a series of states during execution. We make a distinction between the following three states: *enter*, *execution*, and *completion*. A workitem is in the *entered* state when it may be executed, but it is not yet been allocated to a resource. A workitem is the *execution* state, when it has been allocated. A workitem is in the *completed* state, when it has been checked back into the engine, indicating that its execution is completed.

The process associated with the checking in of a workitem is depicted in Figure 6. The thick black lines in the figure shows the paths that can be followed when a request for checking in a workitem arrives at the "check in workitem" place for a given case. Starting from this place it is checked whether: (1) there is a corresponding workitem with the same id in the *executing state* (place "state cases"), (2) the case is active (place "active cases"), and (3)

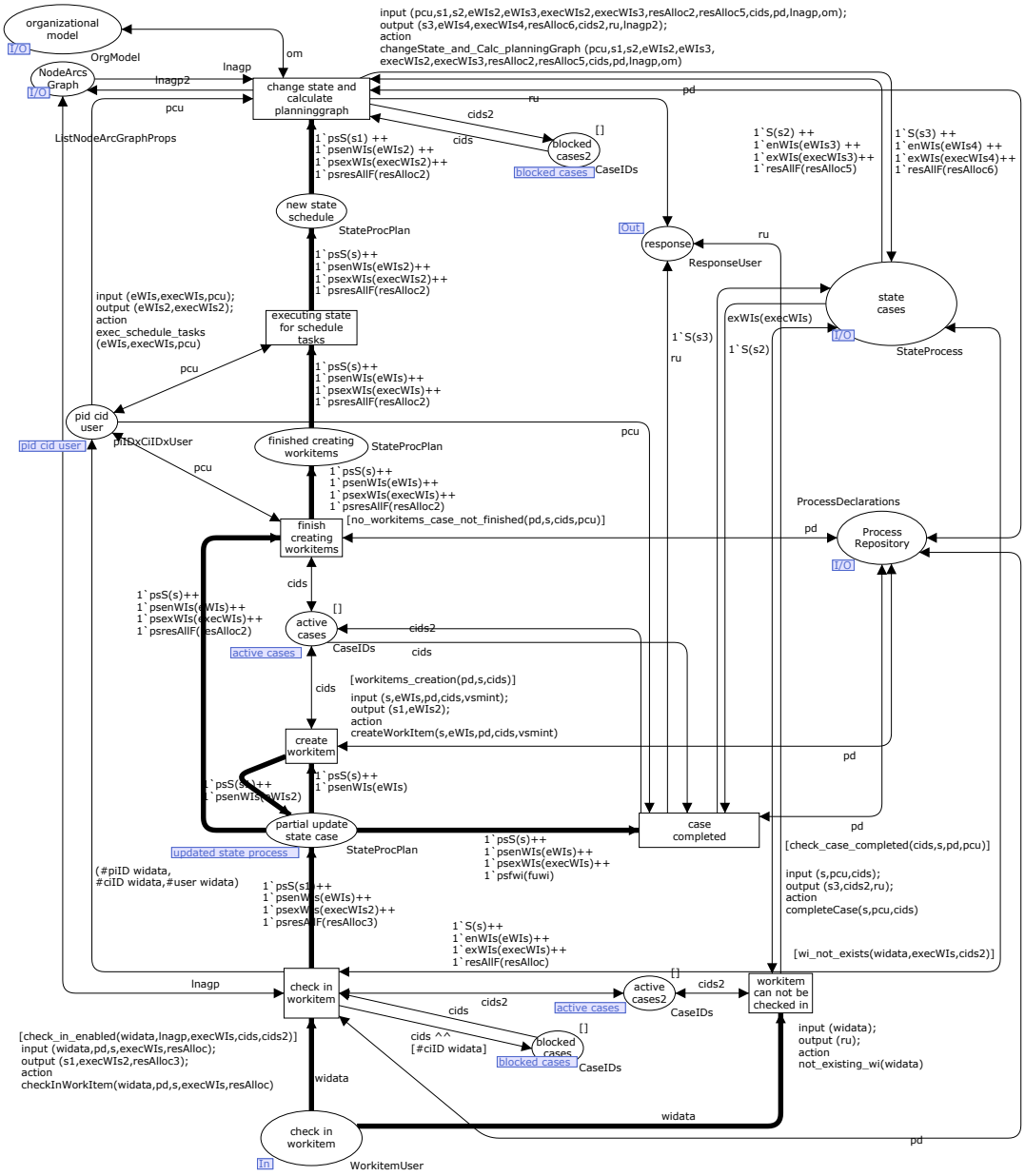

Fig. 6. Checking in of a workitem.

the case is not blocked (place "blocked cases"). If one of the first two prerequisites are not fulfilled, the "workitem can not be checked in" transition will be fired which informs the requester that checking the workitem into the engine was not successful (place "response"). If the prerequisites are fulfilled, the "check in workitem" transition fires and the following actions are taken:

1. the case is blocked (place "blocked cases").
2. the new state of the case is calculated.
3. the resource allocation information for the workitem, being checked into the engine is removed from the state information in the "state cases" place.
4. four tokens are produced in the "partial update state case" place containing the following information: the state of the case, the workitems in *entered* or *executing* state, and the resource allocations for the flow workitems in state *executing*.
5. a token is produced in the "pid cid user" case which contains the *ProcessID*, *CaseID* and the user id of the requester.

Several things can happen now. The "create workitem" transition fires when a workitem can be created for a case. When it fires, the following actions are taken:

1. a workitem is created for the task which will be in the *entered* state.
2. the state of the case, in the "state cases" place, will be updated.

If no workitems can be created for the case, the "finish creating workitems" transition will fire which moves the token to place "finished creating workitems". However, it could also happen that no new workitems can be created because the case is complete. In that situation the "case completed" transition fires and the following actions are taken:

1. all case related information is removed from the "state cases" place.
2. the case is deactivated by removing the case id from the "active cases" place.
3. the requester is informed about case completion by putting a token in the "response" place.

After the "finish creating workitems" transition has fired, two steps remain. The first step relates to the "executing state for schedule tasks" transition. This transition changes the state of the schedule workitems, which just have been created, from *entered* into *executing*. The resource allocation for them is done by the planning service.

The second step relates to the "change state and calculate planning graph" transition. When fired, the following actions are taken:

1. a planning problem is formulated and sent to the planning service via place "NodeArcs-Graph".
2. the updated state of the case and the updated resource allocation for the flow workitems is saved in place "state cases".
3. the case is unblocked ("blocked cases2" place).
4. the requester is informed about the successful completion of the workitem by putting a token in place "response".

4.3 Planning Service

Figure 7 shows the uppermost model of the planning service. Looking back at Figure 3, we can see that this model consists of 21 places and 13 transitions. However, modeling all the required behavior necessitated writing hundreds of lines of ML code which indicates that this component is fairly complex in its behavior.

Three different parts can be distinguished in the model shown in Figure 7. First, at the top, there is the part which is responsible for receiving a planning problem, (re)scheduling tasks if needed, and generating warnings that limited time is left for performing tasks preceding a schedule task. Second, the "cancel case" substitution transition is responsible for removing all

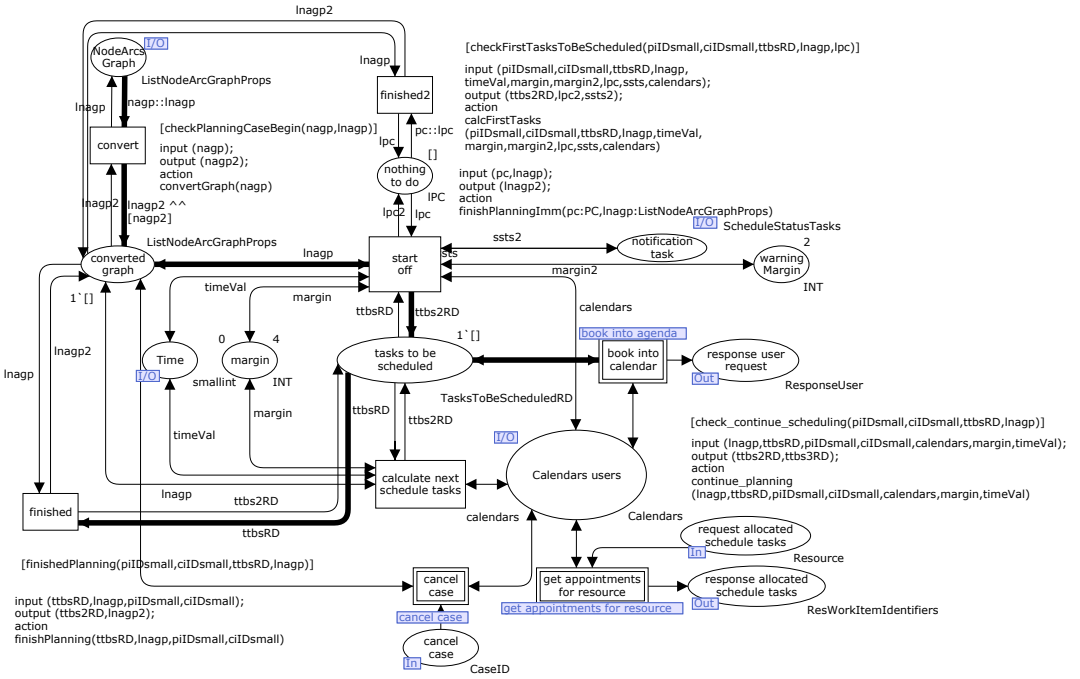

Fig. 7. Top level model of the planning service.

appointments for a case. Third, the "get appointments for resource" substitution transition is responsible for finding all appointments for a specific resource.

For the remaining part of this section we restrict our discussion to the process of receiving a planning problem from the engine and the steps that are taken afterwards. The sequence of these steps are indicated by a path of thick black lines starting from the "NodesArcGraph" place. When a planning problem is sent to the planning service, the required data for the planning problem is added to the "NodesArcGraph" place. The planning problem is represented by a graph containing the planning constraints which hold between the tasks in the corresponding process definition for the case, e.g. the ordering between the tasks. Once this has occurred, the "convert" transition can fire if the planning service is not busy handling another planning problem for the same case. When it fires, nodes are removed from the graph, which represent a task that has already been performed for the case. Also nodes are removed which represent tasks which we are not sure they will ultimately be executed. So, no optimistic planning takes place. The first nodes in the graph, which do not have an incoming arc, represent tasks in the case for which a workitem exists. Note that our algorithm does not take into account any constraints which may hold between tasks that already have been performed and succeeding tasks, which justifies that the nodes for already performed tasks are removed from the graph. However, for more advanced algorithms it might be the case that this removal step is not allowed.

When the "convert" transition fires, a token containing the converted graph is put into the "converted graph" place. Next, the "start off" transition can fire and the following actions are taken:

1. determine whether the first schedule tasks in the graph, viewing it from the start, need to be (re)scheduled or if a warning should be generated. For a user request, the task which is selected for rescheduling is considered to be the first schedule task as only this task needs to rescheduled and possibly subsequent schedule tasks.
2. for the schedule tasks which need to be (re)scheduled, the earliest time is calculated at which they may be executed. This is dependent on any preceding tasks which need to be completed.
3. other relevant information for scheduling the task is determined, such as the defined roles and the duration of the task.
4. for every first schedule task, that is a schedule task in the graph for which no preceding schedule task exists, which needs to be (re)scheduled a token containing the information mentioned above is put in the "tasks to be scheduled" place. An example of such a first schedule task is the "first consult" node in Figure 5.
5. for the first schedule tasks in the graph, that are the schedule tasks in the graph for which no preceding schedule task exists, it is determined whether a warning needs to generated because (too) little time is left for performing preceding tasks. If a warning is needed, a notification is sent to the engine via the "notification task" place. The value for deciding how early such a warning needs to be generated, is stored in the "warning margin" place.
6. for each task that needs to be rescheduled, a notification is sent to the engine via the "notification task" place.

The first tasks which need to be (re)scheduled are added to the "tasks to be scheduled" place, and the substitution transition "book into calendar" is responsible for the actual (re)scheduling. The (re)scheduling is done automatically, which means that there is no user involvement. It should be noted that multiple roles can be specified for a schedule task and that for each role specified only *one* resource may be involved in the actual performance of the task. In the "book into calendar" substitution transition a search is started for the first opportunity that for one resource for every required role an appointment can be booked for the respective workitem. If found, an appointment is booked in the calendars of these resources. If the patient for which the case is performed also needs to be present at the appointment, then this is also taken into account.

However, it can also be that no tasks needs to be (re)scheduled at all. This is determined by the "start off" transition which then puts a token into the " nothing to do" place. Afterwards, the "finished2" transition fires removing the planning problem from the "converted graph" place, indicating that the planning problem has been dealt with.

If all schedule tasks for a case that are present in the "tasks to be scheduled" place are (re)scheduled, then it is checked by the guard of the "calculate next schedule tasks" transition whether succeeding schedule tasks in the planning problem graph need to be (re)scheduled. If the transition fires, the following actions are taken:

1. it is determined which subsequent schedule tasks need to be (re)scheduled.
2. for the schedule tasks which need to be (re)scheduled, the earliest time is determined at which they may be executed.
3. the same relevant information for scheduling the task is determined as when the "start off" transition happens.

For each schedule task which needs to be (re)scheduled, a token containing the information described above is put in the "tasks to be scheduled" place triggering another cycle of (re)scheduling and checking. When no subsequent schedule tasks need to be (re)scheduled,

transition "finished" fires. If this transition fires, the planning problem present in the "converted graph" place is removed, indicating that the planning problem has been dealt with.

4.4 Analysis

A serious drawback that we faced was that no meaningful verification of the CPN model was possible due to its size and complexity. Even more, as an unlimited number of business process models and users can be represented, state space analysis would be impossible. Therefore, we have tested the model by manually simulating a well-chosen set of scenarios. Although this approach revealed several errors, it does not guarantee that the final model is indeed error-free. An example of such a test scenario is that a case is executed from begin to end during which some appointments are rescheduled as consequence of a user action.

5 Implementation

In this section, we will elaborate on the development of a concrete implementation of a workflow management system augmented with planning facilities. First, we elaborate on the architecture of the implemented system, followed by a discussion of its application to a realistic healthcare scenario.

5.1 Architecture

Figure 8 shows the architecture of the system that has been realized. We can see the components and services that are used, and the means by which they interact with each other. The open-source workflow system YAWL has been chosen as the workflow engine [2]. For storing the calendars of users, we selected Microsoft Exchange Server 2007 which offers several interfaces for viewing and manipulating these calendars. Together with this system we could easily use Microsoft Outlook 2003 clients for obtaining a view on an individual users calendar. Moreover, these clients are configured in such a way that they can interact with the YAWL system via an adaptor. By doing so, an Outlook client can act as a full workflow client application. Finally, the planning service is implemented as a Java service which communicates with both YAWL and the Microsoft Exchange Server 2007[4].

The dashed rectangles around the components in Figure 8 indicate how each substitution transition in Figure 4 has been realized. For example, the "workflow engine" substitution transition of Figure 4 has been realized using the YAWL workflow engine and an adaptor which communicates with the workflow client application and the planning service. For implementing the system, we clearly benefitted from the knowledge contained in the CPN model. As the model is a complete specification of the system that needs to be implemented, while abstracting from implementation details, we could immediately start coding from it. Particularly, given the ML-code and the logic, e.g. ordering of transitions, in the CPN model, the code has directly been written. However, if existing third party software could provide the desired functionality of a (part of a) substitution transition, then of course this software is chosen. For example, YAWL has been chosen as workflow engine as it provides the majority of the functionality that

[4] Of course one could argue that for the implementation of the planning service the corresponding part of the CPN model itself could be used. However, pursuing this approach introduces other complex issues like opening and starting a CPN model without opening CPN Tools, communication with external systems, and so on.

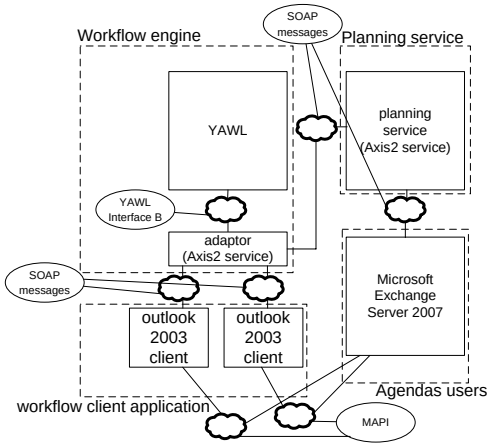

Fig. 8. Architecture of the implemented system. The dashed rectangles indicate how each substitution transition of Figure 4 has been realized. For example, the workflow engine substitution transition has been realized using the YAWL workflow engine together with custom written adaptor.

.

needs to be provided by the "workflow engine" substitution transition. On the other hand, the "planning service" substitution transition has been implemented completely in Java code.

In total, it took around three months for a single person to implement the whole system which involved both component selection and coding. As part of this effort, over 8000 lines of code was written.

5.2 Application

In the remainder of this section, we demonstrate the operation of the system that we realized in the context of a real-life healthcare scenario. As a candidate care process, 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. The healthcare process under consideration is a large process consisting of around 325 activities. 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. For our scenario we will only focus on the initial stages of the process shown in Figure 9.

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 finished before the first visit of the patient (task "first consultation doctor"). For example, the referring hospital needs to be asked to send the radiology data to the AMC (task "call for radiology data"). When the patient visits the outpatient clinic for the first time, the doctor decides whether an "MRI", "CT" or "pre-assessment" or a combination of these tasks is necessary. After performing these diagnostic tests, the results will be discussed during 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 presented model.

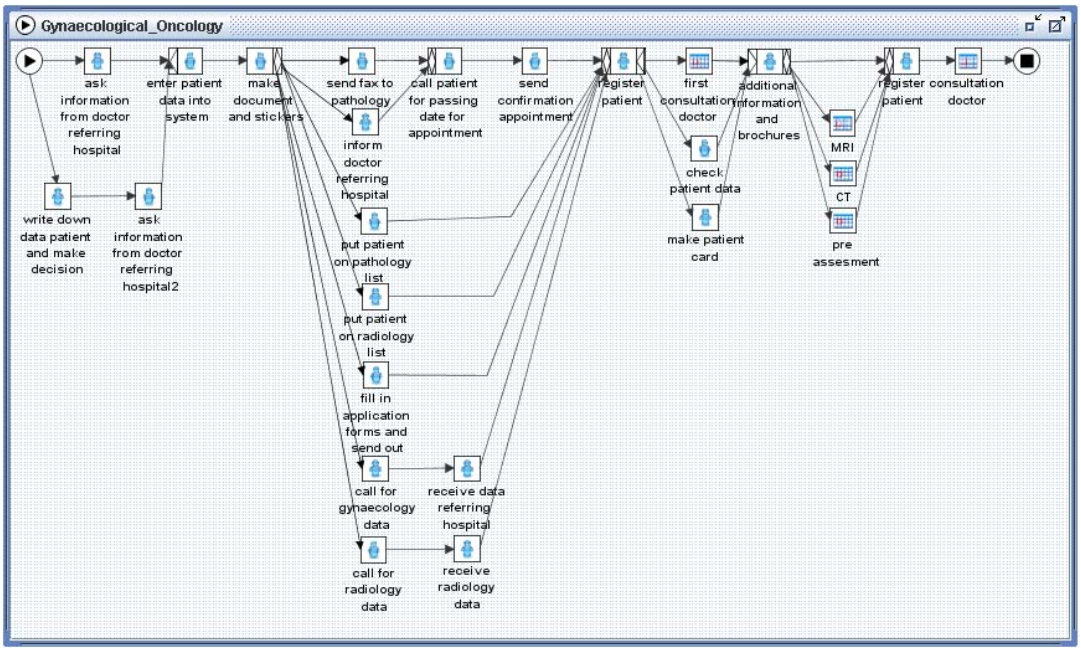

Fig. 9. Screenshot of the YAWL editor showing the initial stages of the gynaecological oncology healthcare process. The flow tasks are indicated by a person icon and the schedule tasks are indicated by a calendar icon. For all schedule tasks, the patient is required to be present.

.

In this scenario, we assume that the task "additional information and brochures" has been performed in which a nurse provides the patient with information and brochures prior to the execution of the diagnostic tests. Furthermore, it has also been confirmed that the doctor requires an MRI and a pre-assessment 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 10. Note that our case has "Oncology" as a process identifier and has "126" as case identifier. Moreover, for the "consultation doctor", "pre assessment", and "MRI" examination, a doctor, an anaesthetist, and MRI machine are needed respectively. Consequently, these tasks have role "doctor", "anaesthetist", and "MRI" respectively. Moreover, the patient is also required to be present.

In Figure 10, going from left to right, we can see that the "MRI" has been scheduled for 8:00 till 8:45, the consultation with the doctor has been scheduled for 13:00 till 13:30 in the calendar of doctor "Nick" and that the pre-assessment has been scheduled for 11:00 till 11:30 in the calendar of anaesthetist "Jules". At the far right, we can see the agenda of patient Fred who also needs to be present during these appointments. All the previously mentioned appointments are also present in his agenda. Moreover, it is important that the "consultation doctor" is scheduled after the "MRI" and "pre-assessment" task, which is also consistent with the corresponding process definition, where the "consultation doctor" task also occurs after them.

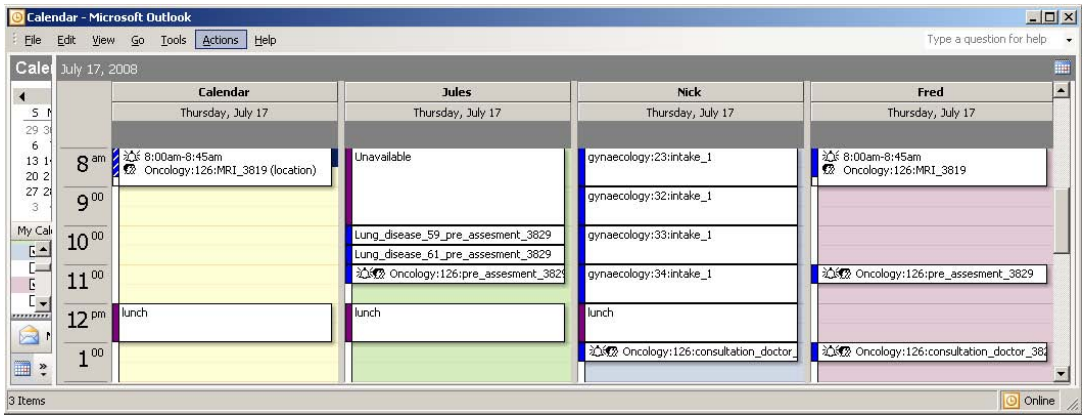

Fig. 10. Screenshot of the calendars for the MRI, consultation with doctor "Nick", and the pre-assessment done by anaesthetist "Jules".

.

However, a problem is now identified: the patient now has been scheduled for an MRI in which they have to lie in a tube. Unfortunately, the patient is suffering from claustrophobia which means that the patient can only be scanned in an MRI having an open system design, a so-called "open-MRI". As the role "MRI" includes both the open and closed MRI, we need to reschedule the patient to use the open-MRI by rejecting the current appointment for the (closed) MRI. Rejecting the appointment means that the appointment must be rescheduled and that the resource who rejected the appointment may not be involved anymore. The effect of this specific rescheduling request can be seen in Figure 11.

In this figure, the messagebox indicates that the MRI has been successfully rescheduled. Moreover, the calendar of the open MRI is now shown on the right hand side. It can be seen, that an appointment for the open MRI has been made, taking place from 14:00 to 14:45. However, as can be seen in the second column of the calendar, which shows the calendar of doctor "Nick", it was also necessary to reschedule the appointment with the doctor which will now occur from 15:00 to 15:30. These changes are also reflected in the agenda of patient "Fred", which is shown in the third column. As can be seen in Figure 9, this rescheduling step is necessary as the task "consultation doctor" occurs after the "MRI" task and the "register patient" task falls in between these two tasks and takes 15 minutes. Moreover, in the left top of the figure, we see the form that is generated automatically for performing the "MRI" task.

6 Related Work

A review of relevant literature shows that extensive research has been done into the problem of appointment scheduling in healthcare, e.g. in areas such as appointment scheduling for outpatient service services [6], operating room scheduling [5] and diagnostic resources. However, these approaches tend to focus on specific facilities and not on the complete careflow process. Another approach is described in [22] in which the online problem of scheduling multiple appointments on a single day is considered. In our approach, the whole careflow is taken into account and appointments are scheduled when it is clear that they need to be executed.

Much effort has been put into experimenting and developing workflow management systems so that they can be applied in the healthcare domain [18, 14]. These efforts vary in the

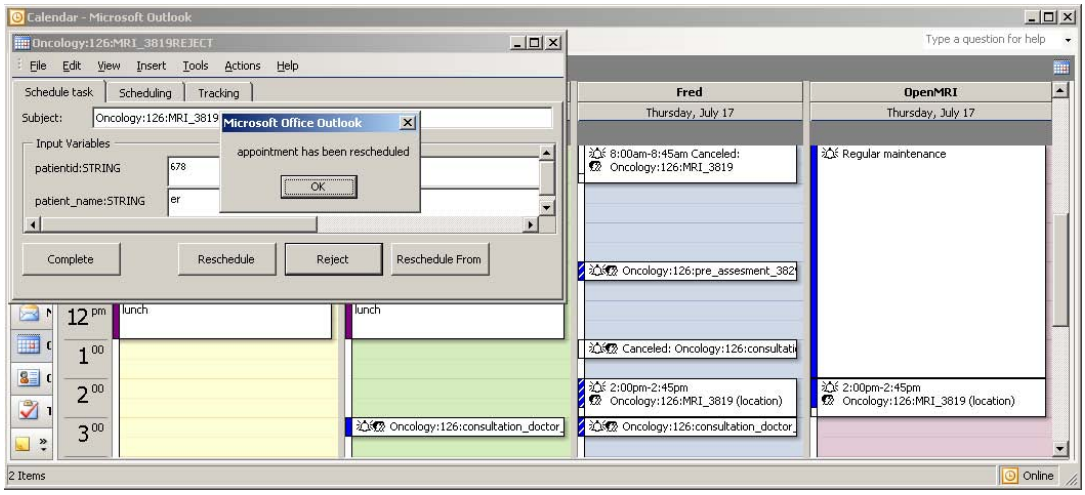

Fig. 11. Result after rescheduling the "MRI" task.

.

sense that they support evidence-based medical procedures, therapies and hospital administrations [16, 15, 21, 4]. One of the most important challenges that needs to be addressed is flexibility support [20]. Unfortunately, current workflow management systems are falling short in providing flexibility [10, 11] which seriously hampers the application of workflow technology in the healthcare domain. In addition to this, support is needed for the cross-departmental nature of healthcare processes [12]. Currently, administrative workflows are typically limited to single departments [19]. Successful implementation of workflow management exists but widespread adoption and dissemination is the exception rather than the rule [14]. It is expected that the use of workflow technology by healthcare institutions will grow dramatically in the future [14] and it is likely that it will become a core component in future healthcare systems [7].

Despite all these efforts, no work has been performed on the combination of appointment scheduling and workflow management systems, i.e., existing approaches are either focusing on planning with little consideration for workflow aspects or are focusing on workflow while ignoring that much work is done via appointments rather than worklists. We are not aware of any research looking at the mixture of flow and schedule tasks.

For various systems, CPNs have been used to formalize and validate functional requirements. For example, the formalization of the design of the so-called worklet service, which adds flexibility and exception handling capabilities to the YAWL workflow system [3], formalizing the implementation of a healthcare process in a workflow management system [13], and presenting a model-based approach to requirements engineering for reactive systems, in which CPNs are used for validating the functional requirements [8]. Related to this is [17], in which CPNs are used for specifying the operational semantics of *new*YAWL, a business process modeling language founded on the well-known workflow patterns[5].

[5] For more information about workflow patterns see http://www.workflowpatterns.com

7 Experiences and Conclusions

In this paper, we have discussed the design and implementation of a workflow management system offering planning and monitoring facilities. As approach, we started with a workflow language, followed by a conceptual model in CPNs and finally a concrete implementation of the system. The conceptual model consists of 27 distinct nets, 377 transitions, 169 places and over 1000 lines of ML code. The construction of the whole model took around three months of work. These figures indicate that a workflow system augmented with planning facilities is a fairly complex system and the task of developing it is far from trivial.

One of the main benefits of building the conceptual model in CPNs is that it can be executed in the CPN Tools offering. In this way, it allows for experimentation during which comprehensive insights can be obtained about the design and behavior of the system to be realized which probably would not have been possible to obtain by pursuing other approaches to designing the system. Parts of the system can be tested early in the development process, thus enabling early detection of design errors. The costs of repairing these errors in this phase of the development process is far less than would be the case in a later phase. For example, when experimenting with the subnet of the planning service we identified errors with regard to the correct planning of appointments.

Another advantage of modeling the conceptual model in CPNs is that it completely specifies the behavior of the system to be implemented while abstracting from implementation details and language specific issues. So, for the conceptual model we only needed to worry about the behavior of the system, while for the implementation we focused on the realization. In this way, these kinds of issues are distinguished, allowing for a separation of concerns. The importance of this distinction can probably best be illustrated by the fact that it took more than 3 months to build the conceptual model, and just 3 months to implement the whole system. For the implementation of the system it was necessary to produce over 8000 lines of code by hand. Although the main functionality of the system was fully implemented during the implementation phase, a significant amount of time still needs to be spent on component selection, coding, and dealing with residual implementation issues.

The fact that we started completely from scratch ending up with a concrete implementation of the system with the proposed functionality shows both the applicability and feasibility of our approach. However, the developed system has only been tested in a limited set of scenarios. As future work, we plan to systematically test parts of the system by "replacing" one or more components in the conceptual model by a complete implementation for it, based on third party software, allowing for the testing of thousands of scenarios. In this way by simply executing the CPN model, we are able to identify errors in the components which probably would not have been found with using a scenario based approach of testing. In addition to this, we plan to use the conceptual model for evaluating alternative planning approaches using various performance indicators.

References

1. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2002.
2. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
3. M.J. Adams. *Facilitating Dynamic Flexibility and Exception Handling for Workflows.* PhD thesis, Faculty of Information Technology, Queensland University of Technology, 2007.

4. L. Ardissono, A.D. Leva, G. Petrone, M. Segnan, and M. Sonnessa. Adaptive medical workflow management for a context-dependent home healthcare assistance service. *Electronic Notes in Theoretical Computer Science*, 146(1):59–68, 2006.
5. 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.
6. T. Cayirli and E. Veral. Outpatient scheduling in health care: a review of literature. *Product Operations Management*, 12(4):519–549, 2003.
7. A. Dwivedi, R. Bali, A. James, and R. Naguib. Workflow Management Systems: the Healthcare Technology of the Future? In *the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 4, pages 3887–3890, 2001.
8. J.M. Fernandes, S. Tjell, and J.B. Jorgensen. Requirements Engineering for Reactive Systems with Coloured Petri Nets: the Gas Pump Controller Example. In K. Jensen, editor, *Proceedings of the Eight Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 207–222, 2007.
9. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *STTT*, 9(3-4):213–254, 2007.
10. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Adaptive Workflow Systems*, Special Issue of Computer Supported Cooperative Work, 2000.
11. R. Lenz, T. Elstner, H. Siegele, and K. Kuhn. A Practical Approach to Process Support in Health Information Systems. *JAMIA*, 9(6):571–585, December 2002.
12. R. Lenz and M. Reichert. IT Support for Healthcare Processes - Premises, Challenges, Perspectives. *Data and Knowledge Engineering*, 61:49–58, 2007.
13. R.S. Mans, W.M.P. van der Aalst, P.J.M. Bakker, A.J. Moleman, K.B. Lassen, and J.B. Jorgensen. From Requirements via Colored Workflow Nets to an Implementation in Several Workflow Systems. In K. Jensen, editor, *Proceedings of the Eight Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 187–206, 2007.
14. M. Murray. Strategies for the Successful Implementation of Workflow Systems within Healthcare: A Cross Case Comparison. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 166–175, 2003.
15. M. Poulymenopoulou and G. Vassilacopoulos. A Web-based Workflow System for Emergency Healthcare. In *Proceedings of the MIE 2002 conference*, 2002.
16. S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible Guideline-based Patient Careflow Systems. *Artificial Intelligence in Medicine*, 22(1):65–80, 2001.
17. N.C. Russell, A.H.M. ter Hofstede, and W.M.P. van der Aalst. newYAWL: specifying a workflow reference language using coloured petri nets. In K. Jensen, editor, *Proceedings of the Eight Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2006)*, volume 584 of *DAIMI*, pages 107–126, Aarhus, Denmark, October 2007. University of Aarhus.
18. G. Russello, C. Dong, and N. Dulay. Consent-Based Workflows for Healthcare Management. In *Proceedings of 2008 IEEE Workshop on Policies for Distributed Systems and Networks (Policy 08)*, pages 153–161, Palisades, NY, US, 2008.
19. X. Song, B. Hwong, G. Matos, and A. Rudorfer. Understanding and classifying requirements for computer-aided healthcare workflows. In *COMPSAC (1)*, pages 137–144. IEEE Computer Society, 2007.
20. M. Stefanelli. Knowledge and Process Management in Health Care Organizations. *Methods Inf Med*, 43:525–535, 2004.
21. S.W. Tu, M.A. Musen, R. Shankar, J. Campbell, K. Hrabak, J. McClay, S.M. Huff, R. McClure, C. Parker, and R. Rocha. Modeling Guidelines for Integration into Clinical Workflow. *Studies in Health Technology and Informatics*, 107:174–178, 2005.
22. I. Vermeulen, H. La Poutré, S.M. Bohte, S.G. Elkhuizen, and P.J. Bakker. Decentralized Online Scheduling of Combination-Appointments in Hospitals. In *Proceedings of ICAPS-2008, the International Conference on Automated Planning and Scheduling*, Sydney, Australia, 2008. AAAI Press.