Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

From Task Descriptions via Colored Petri Nets Towards an Implementation of a New Electronic Patient Record Workflow System

Jens Bæk Jørgensen[1], Kristian Bisgaard Lassen[1], Wil M. P. van der Aalst[2]

[1] Department of Computer Science, University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark. e-mail: {jbj,k.b.lassen}@daimi.au.dk
[2] Department of Mathematics & Computer Science, Eindhoven University of Technology P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands. e-mail: w.m.p.v.d.aalst@tue.nl

The date of receipt and acceptance will be inserted by the editor

Abstract. We consider a given specification of functional requirements for a new electronic patient record system for Fyn County, Denmark. The requirements are expressed as task descriptions, which are informal descriptions of work processes to be supported. We describe how these task descriptions are used as a basis to construct two executable models in the formal modeling language Colored Petri Nets (CPNs). The first CPN model is used as an execution engine for a graphical animation, which constitutes a so-called Executable Use Case (EUC). The EUC is a prototype-like representation of the task descriptions that can help to validate and elicit requirements. The second CPN model is a Colored Workflow Net (CWN). The CWN is derived from the EUC. Together, the EUC and the CWN are used to close the gap between the given requirements specification and the realization of these requirements with the help of an IT system. We demonstrate how the CWN can be translated into the YAWL workflow language, thus resulting in an operational IT system.

Keywords: Workflow Management, Executable Use Cases, Colored Petri Nets, YAWL.

1 Introduction

In this paper, we consider how to come from a specification of user requirements to a realization of these requirements with the help of an IT system.

Our starting point is a requirements specification for a new Electronic Patient Record (EPR) system for Fyn County [11] that existed in 2006 when this paper was written. Fyn County was one of the thirteen counties in Denmark and was responsible for all hospitals and other health-care organizations in its county. We focus on functional requirements for the new EPR system for Fyn County; specifically, we look at seven work processes that must be supported. The work processes cover what can happen from the moment a patient is considered for treatment at a hospital until the patient is eventually dismissed or dead.

In the requirements specification, these work processes are presented in terms of *task descriptions* [19, 20], in the sense of Lauesen. A task description is an informal, prose description. An essential characteristic of a task description is that it specifies what users and the IT system do together. In contrast to *use cases* [10], the split of work between users and IT system is not determined at this stage. Task descriptions are meant to be used at an early stage in requirements engineering and software development projects.

This means that there is a natural and large gap between a task description and its actual support by an IT system. To help bridging this gap, we propose to use *Colored Petri Nets (CPNs)* [15,16] models. CPNs provide a well-established and well-proven language suitable for describing the behavior of systems with characteristics like concurrency, resource sharing, and synchronization. CPN are well-suited for modeling of workflows or work processes [5]. The CPN language is supported by *CPN Tools* [33], which have been used to create, simulate, and analyze the CPN models that we will present in this paper.

Figure 1 outlines the overall approach to be presented in this paper.

The boxes in the figure present the artifacts that we will consider in this paper. A solid arrow between two nodes means that the artifact represented by the source node is used as basis to construct the artifact represented by the destination node.

The leftmost node represents the given task descriptions. Going from left to right, the next node represents an *Executable Use Case (EUC)* [18], which is a CPN model augmented with a graphical animation. EUCs are formal and executable representations of work processes

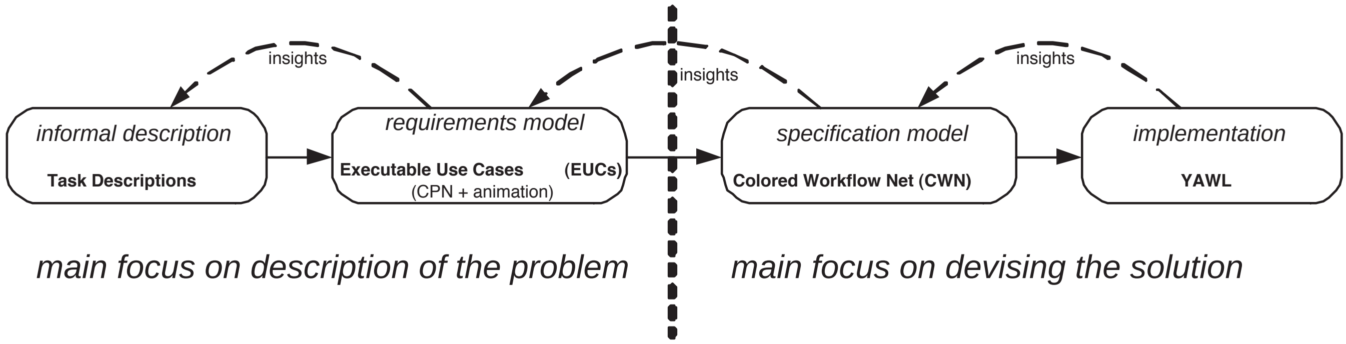

Fig. 1. Overall approach.

to be supported by a new IT system, and can be used in a prototyping fashion to specify, validate, and elicit requirements. The node *Colored Workflow Net* (CWN) represents a CPN model, derived from the EUC CPN, that is closer to an implementation of the given requirements. The rightmost node represents the realization of the IT system itself. In this case study, a prototype has been developed using the YAWL workflow management system [1].

The vertical line in the middle of the figure marks a significant division between "analysis artifacts" to the left and "design and implementation artifacts" to the right. The analysis artifacts represent descriptions of the problems to be solved, in the form of specifying the core work processes that must be supported by the new IT system. To the left of the line, the focus is on describing the problems, not on devising solutions to these problems. In particular, to the left of the line, it is not specified what we want the new IT system itself to do. The arrow between the nodes Executable Use Cases and Colored Workflow Nets represents the transition from analysis, in the form of describing the problem, to design, in the form of devising the solution.

It should be noted that we are not advocating any particular kind of development process in this paper. Figure 1 should not be read to imply that we are proposing waterfall development. In real projects there will often be iterations back and forth between the artifacts in consideration, as is indicated by the dashed arrows. Also, we do not advocate that CPN is the only language that can be used for this approach either. We use CPN since we are comfortable with the language and it provides the tool support needed for the development process, but other languages could be used as well.

It is important to note that it is not possible to use YAWL already in the early phases of the approach. YAWL is implementation oriented while a CPN is more conceptual abstracting from issues such as persistence, form generation, etc. CPN Tools can be used for simulation and animation, but not for enacting real workflows. Therefore, a system like YAWL is needed. However, already using a system like YAWL in the early phases is not advisable since it limits the generation of requirements which do not fit the workflow paradigm. In fact, it should also be possible to decide not to use a workflow system based on the EUCs generated. All of this justifies the use of two languages (CPN and YAWL).

The case study presented in this paper is used to illustrate Figure 1. It has been taken from the medical domain. As pointed out in [25, 26] "careflow systems" pose particular requirements on workflow technology, e.g., in terms of flexibility. Classical workflow-based approaches typically result in systems that restrict users too much. As will be shown in this paper, task descriptions aim at avoiding undesired restrictions. Moreover, the state-based nature of CPNs and YAWL allows for more flexibility than conventional event-based systems, e.g., using the *deferred choice pattern* [2], choices can be resolved by the health-care workers (rather than a decision by the system).

This paper is related to one of our previous publications [3] where we used a similar approach based on EUCs and CWNs. In the earlier work, however, we considered a different domain, namely banking, we did not consider task descriptions, and we used BPEL as target language instead of YAWL.

The emphasis in this paper is on the different stages of the software development and how these can be glued together. We did not have the opportunity to involve real stakeholders in the work we report on.

This paper is structured as follows: Section 2 is about task descriptions, both in general and about the specific task description we will use as case study. Section 3 is about Executable Use Cases (EUCs). In Section 4, we describe the Colored Workflow Net (CWN). Section 5 considers the realization of the EPR system in YAWL. Related work is discussed in Section 6 and the conclusions are drawn in Section 7.

2 Task Descriptions

In this section, we first present task descriptions in general and then we introduce the specific task descriptions for Fyn County's Electronic Patient Record (EPR) that we will focus on in this paper. Finally, we motivate why we move from task descriptions only to EUCs rather than directly implementing the system.

2.1 Task Descriptions in General

In this context, a *task* is a unit of work that must be accomplished by users and an IT system together. A task forms a unit in the sense that after having completed a task, it will feel natural for the user to take a break. Tasks may be split into *subtasks*. An example of a subtask is "register patient".

The *descriptions of subtasks* in a task description are on the left side of the dividing line in Figure 1. A task description, however, may also contain proposals about how to support the given subtasks. *Solution proposals* constitute descriptions, which are to the right of the dividing line in Figure 1. The explicit division into subtasks and solution proposals enforces a strict split between describing a problem and proposing a solution. With solution proposals, the description then properly changes name to a *Task and Support description*. A solution proposal for the subtask "register patient" could be "transfer data electronically from own doctor".

Variants in task description are used to specify special cases in a subtask. Instead of writing a complex subtasks, [20] suggests to extract the special cases in variants, making the subtasks and variants easier to read.

2.2 Task Descriptions for Fyn County's EPR

The task descriptions for Fyn County's EPR that we consider are the following:

1. Request before patient arrives
2. Patient arrives without prior appointment
3. Reception according to appointment
4. Mobile clinical session
5. Stationary clinical session
6. Terminate course of events
7. Patient dies

Descriptions for each of these seven work processes are given in [11] (in Danish). In this paper, we will use "Request before patient arrives" to illustrate our approach. This description is translated into English and presented in Tables 1 and 2. As can be seen, it is a Task and Support description. Except from the translation from Danish into English, the description is presented unchanged (which explains the presence of question marks and other peculiarities).

The subtasks in tables 1 and 2 are named 1, 1a, 1b, 1c, 2, etc. The meaning behind this scheme is that a name without any letter suffix, such as subtask 1, is a main subtask, whereas the subtasks 1a, 1b, and 1c, are variants of 1.

2.3 From Task Descriptions to Executable Use Cases

One of the main motivations behind task descriptions is to alleviate some problems related to use cases. A use case describes an interaction between a computer system and one or more external actors. In the sense of Sommerville [29], use cases are effective to capture "interaction viewpoints", but not adequate for "domain requirements". Typically a task description has a broader perspective than a use case, and, as such, is a means to address domain requirements as well.

In a use case description, the split of work between users and the system is determined. In contrast, in a task description, this split of work is not fixed. A task description describes what the user and the system must do together. Deciding who does what is done at a later stage. Thus, a task description can help to avoid making premature design decisions. In other words, a task description is a means to help users to keep focus on their domain and the problems to be solved, instead of drifting into designing solutions of sometimes ill-defined and badly understood problems.

On the other hand, use cases and task descriptions share the salient characteristics that they are static descriptions: They are mainly prose text (may be structured or semi-structured) possibly supplemented with some drawings, e.g., containing ellipses, boxes, stick men, and arrows as in UML use case diagrams. Both task descriptions and use cases may be read, inspected, and discussed, and in this way, they may be improved.

A traditional prototype, though, tends to focus on an IT system itself, in particular on that system's GUI, more than explicitly on the work processes to be supported by the new IT systems. This has been a main motivation to introduce EUCs as a means to be used in requirements engineering. It is meant to provide executable descriptions of new work processes and possibly of their intended computer support, and in this way, be able to talk back to the user, thereby facilitating discussions about both work processes and IT systems support.

3 Executable Use Cases (EUCs)

In this section, we first present EUCs in general and then we introduce the specific EUC related to Fyn County's EPR that we will focus on in this paper. We also consider how to come from EUCs to CWNs.

3.1 Executable Use Cases in General

An EUC consists of three tiers, as indicated in Figure 2.

Each tier represents the considered work processes that must be supported by a new system. The tiers use different representations: Tier 1 (the *informal tier*) is an informal description; Tier 2 (the *formal tier*) is a formal,

Table 1. Task description: Request before patient arrives (First part - to be continued in Table 2).

Task 1: Request before patient arrives		
Establish episode of care or continue the establishment process if it had been parked or transferred. The request can involve a clinical session where the episode of care is refined before the patient arrives. **Start** Request from the patient's practitioner, specialist doctor, other hospital, or authority. Request can also be supplementary information that were missing previously, or when the task was transferred to another person (e.g. from the secretary to the doctor). **End** When the episode of care is established/adjusted and the patient called in or added to the waiting list. **Frequency** Per user: ??. For the whole hospital: ??. **Critical situations** **Users** The secretary is the immediate user, but the task can be transferred to others.		
Subtask and variant number	Subtask	Solution proposal
1.	Register patient. (See data description)	Transfer data electronically from the patients doctor, etc. (Medcom)
1a.	Patient exist in system. Update data	
1b.	Healthy partner must be enrolled	??
1c.	Personal security ??	??
2.	Establish episode of care and register data, i.e., the preliminary diagnosis. (See data description, including support in use of SKS classification.)	Transfer data electronically from own doctor, etc. (Medcom)
2a.	Problem: Diverging code systems and structures in the electronic messages.	Support the manual transfer of data from the electronic data form to the system form.
2b.	Episode of care is already established. Data may need to be adjusted, e.g., date of patient appointment.	

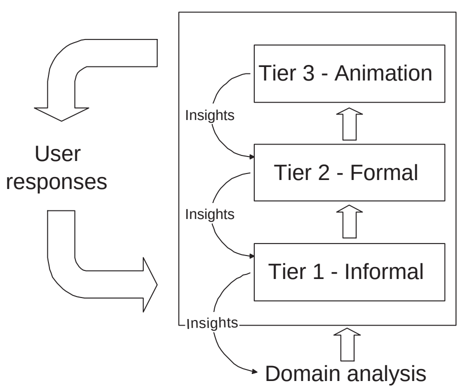

Fig. 2. Executable Use Cases.

executable model; Tier 3 (the *animation tier*) is a graphical animation of Tier 2, which uses only concepts and terminology that are familiar to and understandable for the future users of the new system.

As indicated by Figure 2, the three tiers of an EUC should be created and executed in an iterative fashion. The first version of Tier 1 is based on domain analysis, and the first version of tiers 2 and 3, respectively, is based on the tier immediately below.

The formal tier of an EUC may in general be created in a number of formal modeling languages. We have chosen CPN because we have good experience with this language and its tool support, but other researchers and practitioners may have other preferences, e.g., other options could be statecharts [13], UML activity diagrams [22], or other dialects of Petri nets than CPN.

Table 2. Task description: Request before patient arrives (Second part - continued from Table 1).

Task 1: Request before patient arrives		
Subtask and variant number	Subtask	Solution proposal
2c.	Problem: The patient can concurrently be involved in other episodes of care and be enrolled more places and in more departments. It can be hard to get an overview of who has the nursing responsibility and who is providing a bed. Also, there may be a need to see previous episode of care, given that the patient agrees.	
3.	Possible clinical session to plan the episode of care (e.g. if the establishment process is transferred to a doctor).	
4.	Print patient call-up (or other form of call-up).	
4a.	Patient is transferred to the waiting list	
4b.	Information is missing and the task is parked with time monitoring	
4c.	The case is transferred to another, perhaps with time monitoring.	
4d.	The request is possibly denied.	
5.	Request interpreter for the time of admission.	

As was mentioned in Section 2.3, EUCs have notable similarities with traditional high-fidelity prototypes of IT systems; this comparison is made in more detail in [9]. In [17], it is described how an EUC can be used to link and ensure consistency between, in the sense of Jackson [14], user-level requirements and technical software specifications. Jackson's division into requirements and specifications resembles the division into subtasks and solution proposals in task descriptions. User-level requirements and subtasks lie to the left of the dividing line in Figure 1; technical software specifications and solution proposals lie to the right.

Like a task description, an EUC can have a broader scope than a traditional use case. The latter is a description of a sequence of interactions between external actors and a system that happens at the interface of the system. An EUC can go further into the environment of the system and also describe potentially relevant behavior in the environment that does not happen at the interface. Moreover, an EUC does not necessarily fully specify which parts of the considered work processes will remain manual, which will be supported by the new system, and which will be entirely automated by the new system. An EUC can be similar to, indeed, a task description. Thus, despite the name, EUCs can be more similar to task descriptions than to use cases. The name "executable use cases" was originally chosen to make it easy to explain the main idea of our requirements engineering approach to people, who were already familiar with traditional prose use cases.

3.2 Executable Use Case for Fyn County's EPR

We have made an EUC that covers all seven task descriptions listed in the beginning of Section 2.2.

In this section, we will present the part of the EUC that corresponds to the task description of Tables 1 and 2, where we have focussed on the subtasks and not on the solution proposals. The informal tier of the EUC is the task description itself.

An extract of the formal tier is shown in Figure 3; this figure presents the CPN model that corresponds to the task description from Tables 1 and 2.

In [20], it is said that the subtasks in a task description can be done in any interleaving and any number of times. We identify that often some partial order of subtasks exists that reflects a "normal execution" for a particular task. The term normal execution is to be understood as the most common or the ordinary way a task would be carried out. Therefore, the model in Figure 3 for Task 1, is not just one place with all subtasks linked to this place, describing the situation that [20] proposes. Instead we enforce a process orientation of the task to indicate that under normal circumstances the EPR system would, e.g., execute the subtask `Register patient` before the subtask `Establish episode of care`.

We have annotated the model by *thick lines* to denote main path of the task description, i.e., where the subtasks are carried out consecutively; e.g. to go from the place `Ready to make appointment` to `Patient ready for arrival`. *Solid lines* denote subtasks and variants of subtasks. *Dashed lines* denote added structure to the

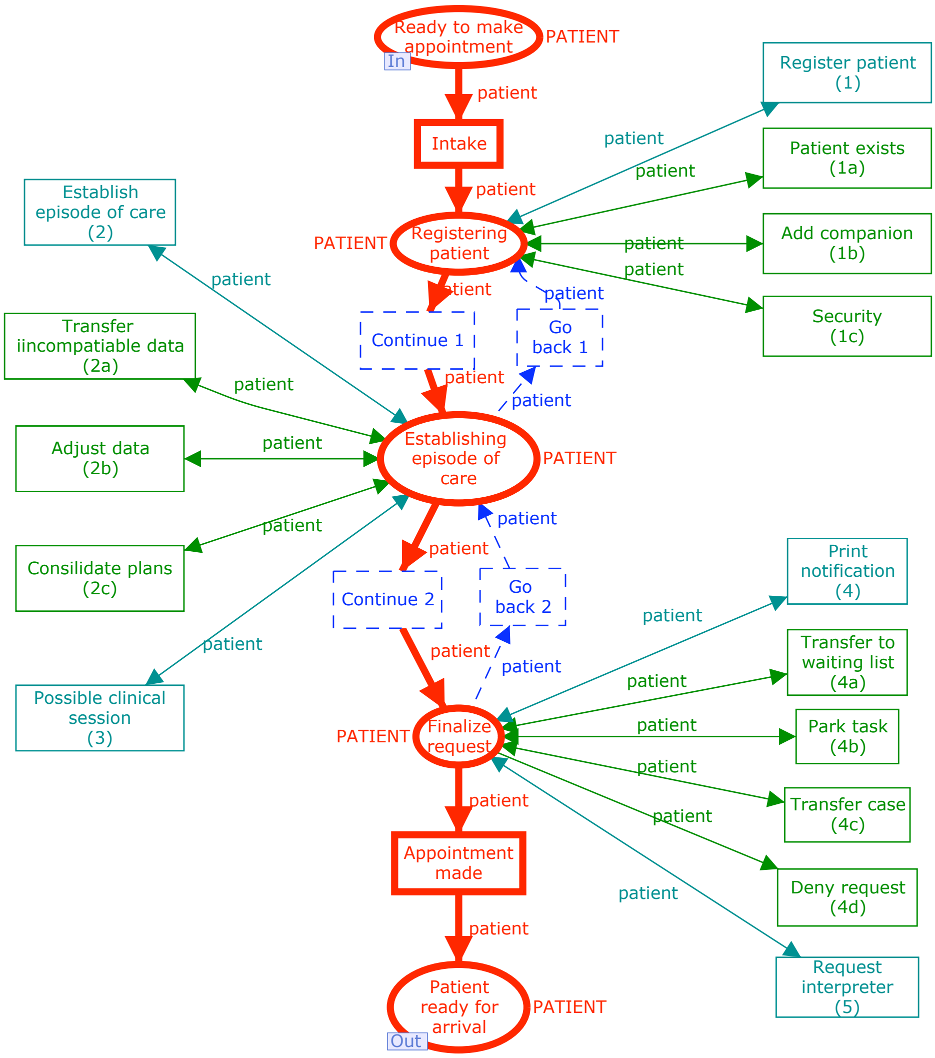

Fig. 3. Task 1 modeled in CPN

model to assert that desired interleavings of subtasks (or variants) are possible.

The behavior of model for Task 1, is that any of the subtasks can be executed in any order, since none of the `Continue` or `Go back` transitions are guarded. If we wanted to enforce that particular circumstances must be fulfilled before proceeding from, e.g., the place `Registering patient` to the place `Establishing episode of care`, we could add a guard to `Continue 1` stating this requirement. This guard could e.g. express that the subtask `Register patient` must be performed at least once.

In Figure 4, we outline how the formal and animation tiers are related. At the bottom, we see the formal tier executing in CPN Tools. Please note that the shown module of the CPN model contains seven transitions (the rectangles), and that each of these transitions corresponds to one of the considered tasks (cf. the list in the beginning of Section 2.2). At the top is the animation tier in BRITNeY [32,31], the animation facility of CPN Tools. The two tiers are connected by adding animation drawing primitives to transitions in the CPN model. These primitives update the animation.

The animation tier is a view on the state of, and actions in the formal tier. When a transition occurs in the formal model it is reflected by updates to the animation tier. Therefore, the behaviors of the two tiers remain synchronized.

Using the animation tier the user can interact with each of the seven tasks. Within the animation of each task, subtasks can be selected and executed. When a subtasks is chosen for execution, the animation user can see visually what is happening and see which persons and/or what devices are involved in completing the subtask. In the snapshot shown in Figure 4, the animation visualizes Task 1. It shows that the animation user has chosen to execute subtasks 1, 3, 4, and is about to execute Subtask 4a. We also see that Subtask 4a involves a computer and a secretary. Even tough the main focus of our EUC is on a description of the problem, it also adds suggestion to solve the problem that the task descriptions did not describe. Notice that when we go from left

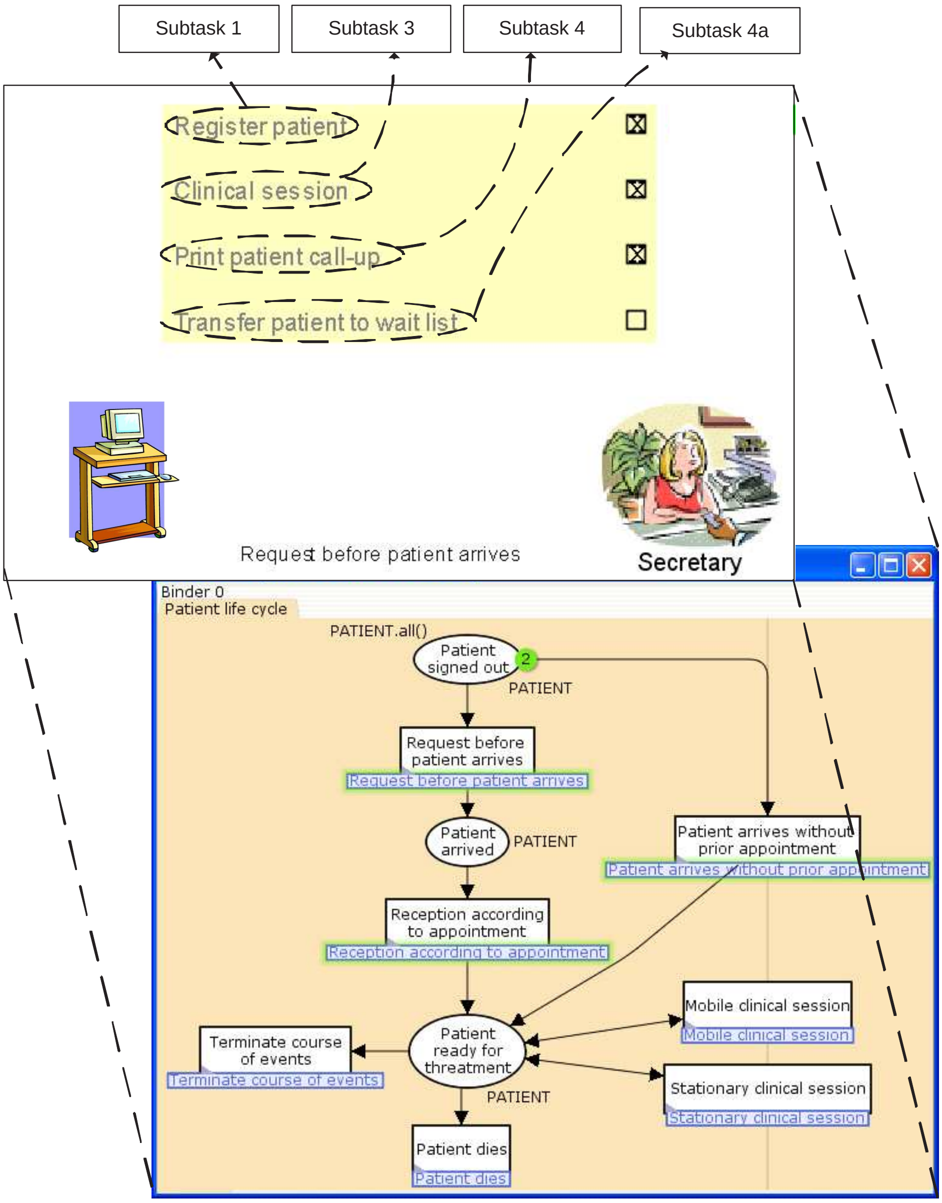

Fig. 4. Connection between animation and formal layer

to right in Figure 1 we gradually add solution proposals to the artifacts.

In the task description in Tables 1 and 2, it was not mentioned, who does what. It is us, the creators of the EUC (software people), who have interpreted the subtasks in this way, i.e., described who does what and what a normal execution of a task is. When showing this animation to the staff at a hospital in Fyns County, we are likely to get more feedback on our interpretations of their daily work than we could get with the static task descriptions only.

3.3 From Executable Use Cases to Colored Workflow Nets

The EUC we have presented above describes real-world work processes at a hospital. When these work processes are to be supported by a new IT system, of course, what goes on inside that system is highly related to what goes on in the real world.

In the approach of this paper (cf. Figure 1), we make separate models of real-world work processes at a hospital (the EUC) and the IT system that must support these work processes (the CWN). This is done to clearly distinguish between the real world, on one hand, and the software, on the other hand. This distinction is advocated by a number of software experts, see, e.g., [14]. Not making this distinction may cause serious confusion.

It is our belief that by building a CWN of the system, rather than implementing it in YAWL directly, we actually make the implementation process much easier. Note that the fact that the EUC and the CWN use the same language (CPN) and tool (CPN Tools) makes it easier to smoothly transition from requirements to specification. When we build the CWN we decide how tasks are ordered, which tasks are done by who, and what the life-cycle of a case should be. These issues would eventually have to be resolved in the final system, but by dealing with them, when building the CWN, we keep a clean distinction between implementation specific issues and these more high-level problems that the CWN addresses. This is essentially the reason why we need the CWN: To resolve the control-flow, resource, and data/case perspectives, without cluttering our workflow design proposals with low-level implementation specific issues. These will be handled at a later stage when we go from our CWN to the implementation platform.

In this way, the CWN we will now present describes the IT system, and, as we will see, this can be used to automatically generate parts of that system.

4 Colored Workflow Nets

In this section, we first present the language Colored Workflow Nets (CWNs), and then show the model that we build related to the Fyn County's EPR. Finally, we discuss differences between the EUC and the CWN.

4.1 Colored Workflow Nets in General

A *Colored Workflow Net* (CWN) [3] is a CPN. Although both the CWN and the formal tier of the EUC use the same language, there are some notable differences. First of all, the scope of the CWN is limited to the IT system, i.e., only those activities that are supported by the system appear in the model. Second, the CWN covers the *control-flow* perspective, the *resource* perspective, and the *data/case* perspective [5]. In the case study of this paper, the EUC covered partially these perspective, but as we move to the right in Figure 1, it is necessary to fully establish the other perspectives as well. Finally, CWNs are restricted to a subset of the CPN language, i.e., CWNs need to satisfy some syntactical and semantical requirements to allow for the automatic realization in a workflow management system [3].

Although a CWN covers the control-flow, resource, and data/case perspectives, it abstracts from implementation details and language/application specific issues. A CWN should be a CPN with only places of type `Case` or `Resource`. These types are as defined in Table 3. As we stated earlier we say that the scope of the CWN is limited to the IT-system. By using the types defined in Table 3, we restrict the designer to only model a workflow system. All data values in the model will either be cases or resources, and it is not possible, e.g., to have places with doctor or nurse tokens; it is only possible to describe these only as part of cases or resources. What the modeler is able to express in CWN, is similar to what many workflow languages provide, and therefore it becomes easy to implement at a later stage.

Table 3. Places in a CWN need to be of type `Case` or `Resource`

```
colset CaseID = union C:INT;
colset AttName = string;
colset AttValue = string;
colset Attribute = product AttName
                           * AttValue;
colset Attributes = list Attribute;
colset Case = product CaseID
                      * Attributes;
colset ResourceID = union R:INT;
colset Role = string;
colset Roles = list Role;
colset OrgUnit = string;
colset OrgUnits = list OrgUnit;
colset Resource = product ResourceID
                          * Roles
                          * OrgUnits;
```

A token in a place of type `Case` refers to a case and some or all of its attributes. Each case has an ID and a list of attributes. Each attribute has a name and a value. Tokens in a place of type `Resource` represent resources. Each resource has an ID and a list of roles and organizational units. The distribution of resources over roles and organizational units can be used in the allocation of resources.

To better understand the definitions of Table 3, let us take a few small examples: A place that route cases could have the marking:

```
1`(C(1),[("name","Eric"),("age","43")]) ++
1`(C(2),[("name","Jane"),("age","17")])
```

This means that there are two cases, each with their own unique id, namely `C(1)` and `C(2)`, and these cases have attributes `name` and `age`. A resource place may have the marking:

```
1`(R(1),["doctor","manager"],["trauma center"])
++ 1`(R(2),["nurse"],["trauma center"])
```

This describes two resources: The first is a description of a person who is both a doctor and a manager (this may indicate he is a chief surgeon), and this person is associated with the organization trauma center. The second resource is a nurse working at the same trauma center.

For more details on CWNs, we refer to [3].

4.2 Colored Workflow Nets for Fyn County's EPR

Figure 5 shows the CWN for the task `Request before patient arrives`. When comparing this CWN with the

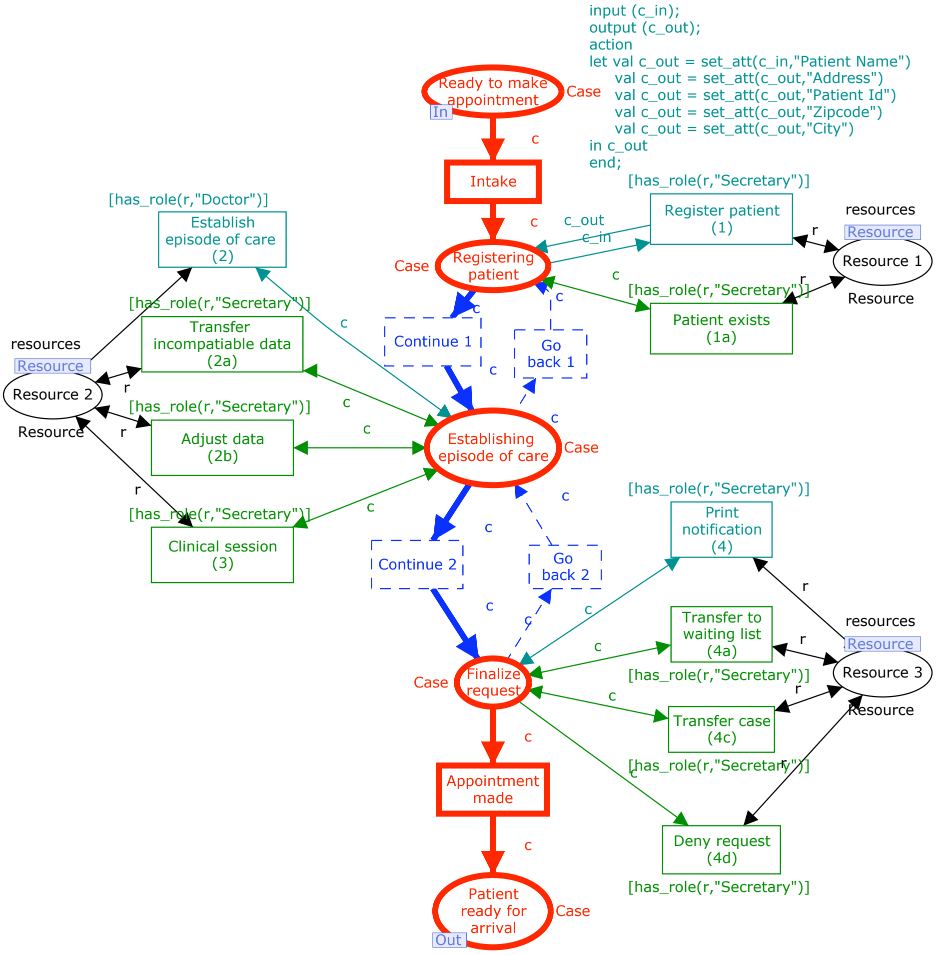

Fig. 5. CWN for the task `Request before patient arrives`

EUC CPN shown in Figure 3, several differences can be observed. First of all, some subtasks shown in the EUC CPN are not included in the CWN because they will not be supported by the IT system. Subtask 1b (`Add companion`) and subtask 2c (`Consolidate plans`) are not included because of this reason. Secondly, Figure 5 includes more explicit references to the resource and data/case perspectives. Note that Figure 5 shows three resource places of type `Resource` defined in Table 3. These resource places hold information on the availability and capabilities of people. Using the concept of a place fusion [15,33], these places together form one logical entity. Places of type `Case` hold information on cases. Cases have several attributes such as `patient name`, `patient id`, `address`, `birth date`, `preliminary diagnosis`, etc. In Figure 5, the relevant attributes are only shown for the task `Register patient`, but, for the sake of readability, not shown for all other tasks[1]; they follow the same form as for `Register patient`.

One of the advantages of using Petri nets is the availability of a wide variety of analysis techniques. In CPN Tools it is possible to simulate models and to do state-space analysis. We have used both facilities. For the state-space analysis we have abstracted from time and color to assess soundness [5]. Initially, we discovered a minor error (a deadlock because we did not connect Subtask 4d properly). However, after repairing this, the CWN was sound. Note that the reachability graph of the CWN shown in Figure 5 for one patient has only 14 nodes and 29 arcs, so it is easy to verify its correctness by hand; e.g. that the process has no deadlocks. For more complicated CWNs, automated state-space analysis of CPN Tools is, however, indispensable to assess correctness before implementation.

4.3 Differences Between the EUC and the CWN

Although the CPN model used in the EUC and the CWN may look alike in structure (cf. figures 3 and 5), they really are two different models. The model used for the EUC is a description of the real world. It describes patients, doctors, nurses, and any other relevant vocabulary for the hospital domain. Its focus is not on how the system could be realized, but simply to describe the work processes that take place in a hospital.

[1] It could be considered to add a further tier to the CWN to hide complex inscription, if it could be justified from a cost-benefit consideration.

On the other hand, the CWN model is a description of the system to support the work processes that were identified in the EUC. In the CWN we do not allow ourselves to use the domain specific vocabulary that we used in the EUC, instead we restrict our vocabulary to only use that which can be supported by a workflow system. This way we end up with a description of a system that is much easier to translate to an actual workflow system. CWNs fit better into the way workflow system are described since they use similar concepts to describe similar things such as cases and resources, whereas the EUC CPN model was build without any restriction on how to express things. This free modeling form results in a EUC CPN model that is harder to map directly into an workflow implementation platform, than a CWN model that has more or less the same concepts as most workflow implementation platforms.

The CWN is a high-level description of the workflow system, in the sense that we do not worry about concrete workflow system implementation details, such as port types, if we where to implement the system in BPEL, or XPath expressions to select the proper parts of a token in an arc expression had we chosen YAWL. These details are all dealt with at a later stage in the implementation of the system, and the CWN high-level description only focuses on the control-flow, how tasks are ordered in the process, the resource perspective, who can do what and with what, and the case perspective, what is the attributes and life-cycle of a case in the system.

Obviously the control-flow in the two models looks very much alike, as mentioned earlier. We think they will often do that in general, since it is the same real world problems that they respectively try to describe and solve - the difference is how they go about it. For example, in Figure 3 the transition `Register patient (1)` is an activity that may occur when a patient is admitted to a hospital, but we do not specify how the activity is actually done, other than we give suggestions to who is involved in doing it when we add the animation layer of the EUC. In Figure 5 there is a transition with the same name, this is, however, not the same behavior that we model here. Here we explicitly say that the activity `Register patient (1)` is a work item that is available to anyone with the role secretary, immediately after the patient workflow is in a state where the patient is ready to be admitted. Also, in the CWN model we specify the data attributes of a case; something we deliberately did not fully establish in the EUC model.

To conclude this section, we summarize the main distinction between EUC and CWN. The CPN model used for the EUC is strictly used to explore what work processes are in the hospital, and the CWN model is used to devise a solution using workflow system concepts, which can be easily mapped into a workflow implementation platform.

5 Realization of the System Using YAWL

In [3], it was shown that for some CWNs it is possible to automatically generate BPEL template code [8]. The *Business Process Execution Language for Web Services* (BPEL4WS or short BPEL) [8] is a textual XML-based language that has been developed to form the "glue" between web services. Although it is an expressive language, it tends to result in models that are difficult to understand and maintain. For example, because of the verbose nature of BPEL code it is not interesting to show it to users (e.g., to visualize management information or to allow for dynamic change [28]). Moreover, BPEL offers little flexibility and no support for the resource perspective.[2] Therefore, we decided to use YAWL [1] rather than BPEL.

YAWL (Yet Another Workflow Language) [1] is based on results achieved by the Workflow Patterns Initiative (`www.workflowpatterns.com`, [2]). YAWL offers direct support for 19 of the 20 patterns identified in [2]. For example, because of its native and unrestricted support of the *deferred choice* pattern [2], it is possible to leave the selection of the next task to the user. This offers more flexibility than BPEL, because it is possible to define for each state what tasks are possible without selecting one (in BPEL this is restricted to the inside of a `pick` activity [8]). Moreover, YAWL also supports the resource perspective (in addition to the control-flow and data perspectives). YAWL is supported by an open source workflow management system that can be downloaded from `www.yawl-system.com`.

Given the fact that YAWL can be seen as a superset of CWNs, it was easy to translate the running example from CPN in YAWL. Figure 6 shows the top-level workflow and the composite task `Request before patient arrives`. Although both models look quite different, a fairly direct mapping was possible from the CWN shown in Figure 5 to the YAWL model shown in Figure 6. All places of type `Case` in Figure 5 are mapped onto conditions in YAWL and transitions in Figure 5 are mapped onto YAWL tasks.[3]

After mapping the CWNs onto a YAWL specification, it is possible to enact the associated workflows. Figure 7 shows a work-list and a form generated by YAWL. The top of the figure shows the work-list of the secretary with user code `secretary4`. It shows work-items associated to three cases. Each of these three cases is in the state `registering` where three tasks are enabled. Therefore, there are 3*3=9 possible work-items. After selecting a work-item related to task `register patient`, three

[2] Note that only recently people started to investigate adding the resource perspective to BPEL, cf. the *WS-BPEL Extension for People (BPEL4People)* initiative `http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/`.

[3] Note that *subtasks* in Task Descriptions correspond to *transitions* in CWNs and *tasks* in YAWL.

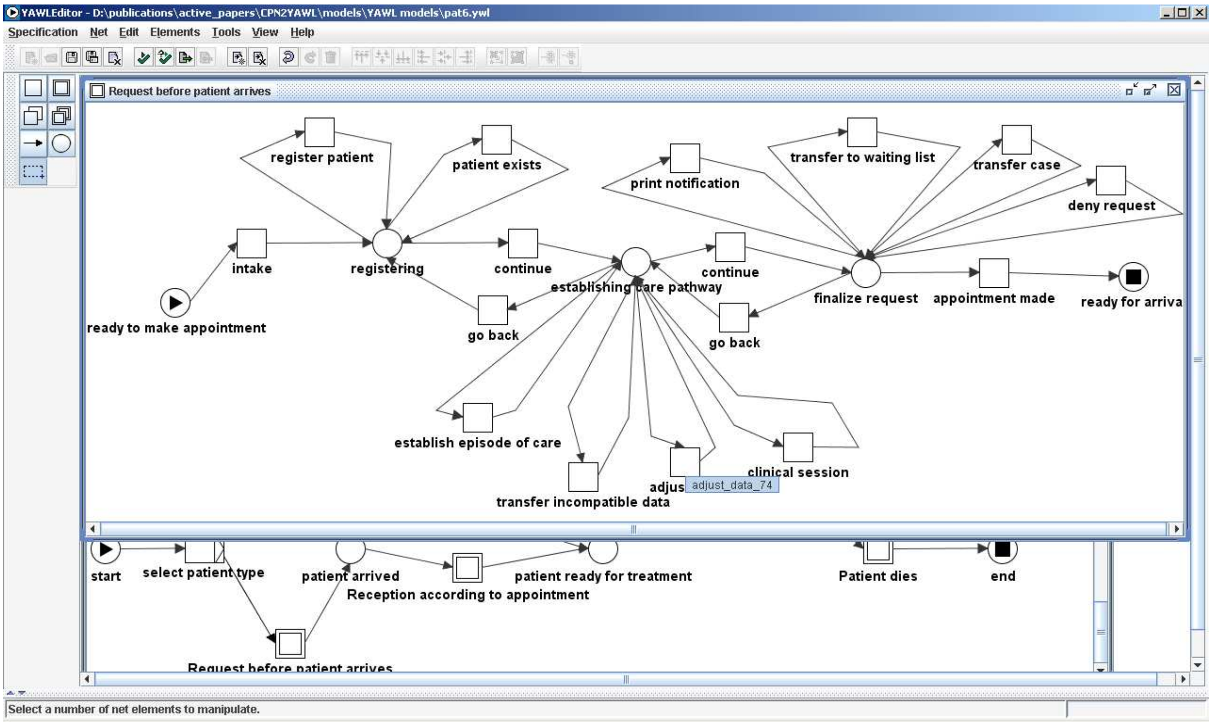

Fig. 6. Screenshot of YAWL editor

work-items disappear from the work-list (the competing tasks become disabled for this patient) and `secretary4` can fill out a form with patient data. After completing the form there are again nine work-items, etc.

The realization of the workflow process in YAWL completes the overall approach shown in Figure 1, i.e., we moved from informal task descriptions, then to EUCs, after that to CWNs, and finally realized the task descriptions in terms of YAWL. Note that, given the availability of a running YAWL system and a CWN, it is possible to construct a running system in a very short period, e.g., in a few hours it is possible to make the process shown in Figure 6 operational. This does include the generation of user forms as shown in Figure 7 but, of course, does not include system integration or the development of dedicated applications. The task of mapping a CWN onto YAWL can be partly automated by using the automatic translation provided by ProM (cf. `www.processmining.org`). ProM is able to automatically map Petri nets in PNML format onto various other formats, including YAWL. However, this translation does not take data and resources into account, so some manual work remains to be done. Nevertheless, it shows that the overall process shown in Figure 1 is feasible. Moreover, we would like to argue that by using our approach initially more time is spent on the requirements, but considerably less time is spent on the actual realization and testing. The intermediate steps (i.e., EUCs and CWNs) enable an efficient implementation.

6 Related Work

This paper builds on the work presented in [3], where we also apply CPN Tools to model EUCs and CWNs. However, in [3], EUCs are not linked to task descriptions and we used BPEL as target language instead of YAWL. The extension with task descriptions was inspired by the work of Lauesen [19,20]. Compared to existing approaches for requirements engineering and use case design [10,12,14,29], our approach puts more emphasis on the two intermediate steps. First of all, we make EUCs with both an animation and formal tier. Second, we use CWNs to link these EUCs to concrete implementations.

Today, workflow technology is used in areas such as radiology [30]. However, there is no systematic and broader support for workflows in health-care organizations. Vendors and researchers are trying to implement "careflow systems" but are often confronted with the need for more flexibility [25,26]. The state concept in CPN and YAWL (e.g., places with multiple outgoing arcs modeling a choice which is resolved by the organization rather than the system) allows for more flexibility than classical workflow systems. We know of one other application of YAWL in the health-care domain. Giorgio Leonardi, Silvana Quaglini et al. from the University of Pavia have used YAWL to build a careflow management system for outpatients [21]. However, they did not use task descriptions, EUCs, and CWNs. Instead they directly implemented the system in YAWL [1].

UML allows a designer to model many different perspectives of a system. There are different types of dia-

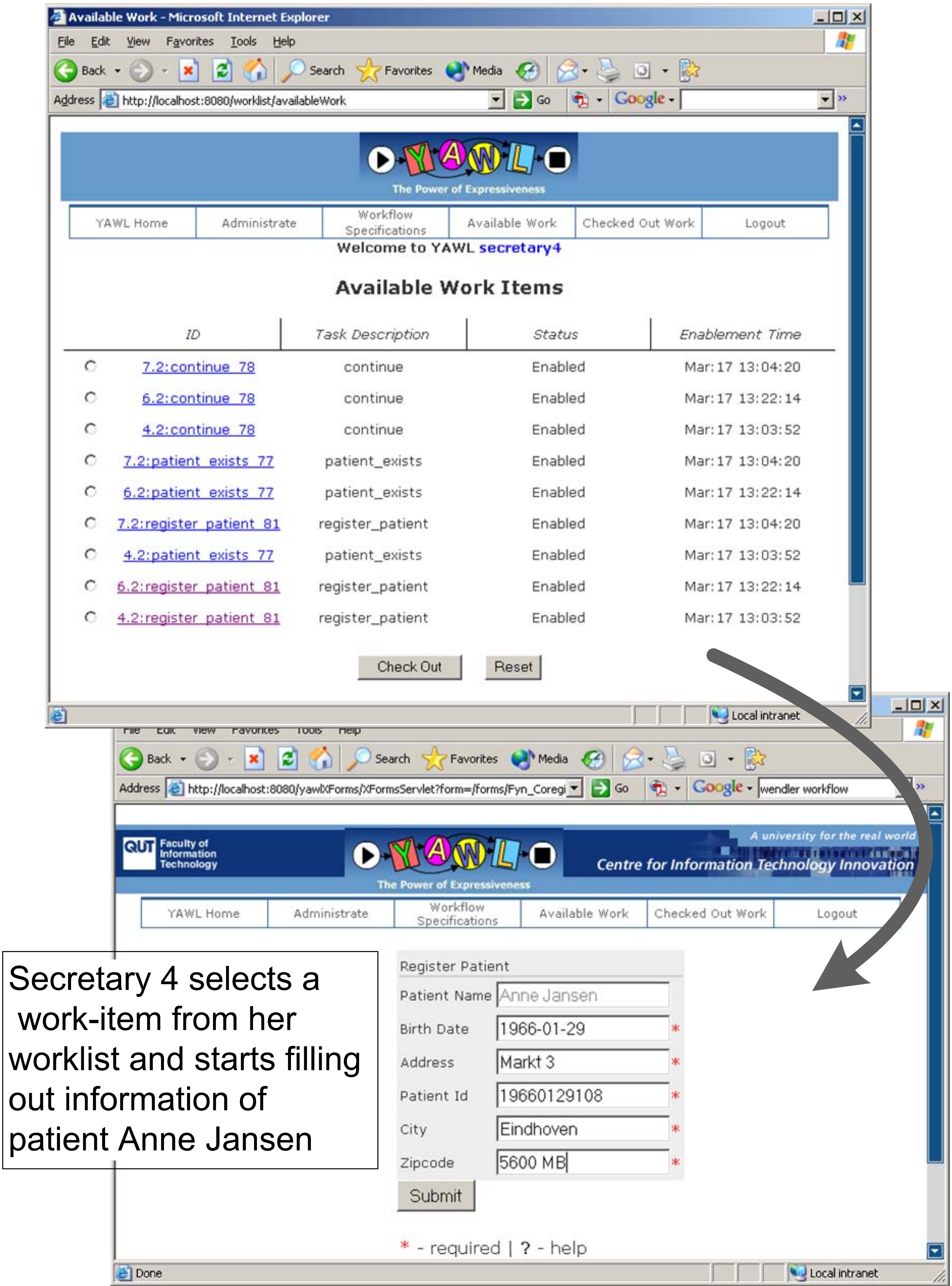

Fig. 7. Screenshot of the YAWL worklist and the form associated to task `register patient`

grams covering different perspectives. For example, the activity diagram and the statechart diagrams focus on the dynamical aspects, while the class diagrams focus on structure and data. Clearly, our approach could be complemented by these aspects. Since the end result of our approach is a running workflow system and not any arbitrary software system, we can limit ourselves to only use the types of models that we have presented in this paper.

7 Conclusions

In this paper, we realized a small careflow system using the four-step approach depicted in Figure 1 and motivated the added value of each of the three transformation steps in our approach. Obviously, the system made using YAWL is not the full EPR for Fyn County. It is just a prototype illustrating the viability of our approach. To come from an extensive and detailed set of task descriptions — as the seven task descriptions we have been considering — to their implementation requires large amounts of work and extensive involvements of the stakeholders. A weakness of the work presented in this paper is the unavailability of stakeholders in coming from the task descriptions to the EUC, the CWN, and the YAWL implementation, i.e., the stakeholders have been extensively involved in the development of the task descriptions but e.g. have not evaluated the YAWL implementation.

We believe that less time needs to be spent on testing the resulting system of our development process in the ideal case where the design has been validated and verified earlier in the EUC. Also, the system is more likely to be accepted by the end-users, since the work processes that they saw and approved in the EUC, reoccur in the final system. It is difficult to empirically prove this. In [27] we report on a study where we measured the effects of introducing workflow technology for 16 processes in six organizations. This study shows how difficult it is to empirically measure the effects of new approaches and systems in business process management.

CPN is the language we used both for Executable Use Cases (EUCs) and Colored Workflow Nets (CWNs). For the actual realization of the system we used YAWL which can be seen as a superset of CWNs (extended with OR-joins and cancellation sets [1]) dedicated towards the implementation of workflows. The state-based nature of these modeling languages fits well with task descriptions, i.e., in a given state it is possible to enable multiple tasks and let the environment select one of these tasks. In many workflow systems, this is not possible, because the systems selects the next step to be executed.

An interesting topic for further research is the consistency between the various models shown Figure 1. In the paper, we assumed that one would go from left to right. However, we are not proposing the waterfall development process and envision that multiple iterations are needed. Moreover, it could be that after some time the system needs to be revised because of new requirements. In such a situation, one would manually need to keep all models consistent. As a result, it is possible that the ECU and CWN are thrown away and that just the YAWL model is updated. To avoid this, it would be good to support the relations and consistency checking among the various artifacts.

Although CPNs and YAWL allow for more flexibility than classical workflow management systems, we would like to argue that in the health-care domain more flexibility is needed than what is provided by YAWL as it has been used in this paper. Work on computer-interpretable guidelines [23] shows that classical workflow languages tend to be too restrictive. Health-care workers should be allowed to deviate and select alternative pathways if needed.

To conclude this paper, we would like to discuss four such extensions to allow for more flexibility.

- **Dynamic change.** The basic idea of dynamic change is to allow for changes while cases are being handled [28]. A change may affect one case (e.g., changing the standard treatment for an individual patient) or many cases (e.g., a new virus forcing a hospital to deviate from standard procedures). Although this approach is very flexible, it requires end-users to be able and willing to change process models.
- **Case handling.** Case handling [6] comprises a set of concepts to enable more flexibility without the need for adapting processes. The basic idea is that there are several mechanisms to deviate from the standard flow, e.g., unless explicitly disabled people can skip and roll-back tasks. Moreover, the control-flow perspective is no longer dominating, i.e., based on the available data the state is constantly re-evaluated and the collection and visualization of data is no longer bound to specific tasks.
- **Worklets.** Worklets [7] allow for the late binding of process fragments, e.g., based on the condition of a patient the appropriate treatment is selected. YAWL supports the uses of worklets, i.e., based on ripple-down rules an appropriate subprocess is selected. The set of ripple-down rules and the repertoire of worklets can be extended on-the-fly thus allowing for a limited form or dynamic change.
- **Declarative languages.** Another way to allow for more flexibility is to use a declarative languages like ConDec [24] or DecSerFlow [4]. Classical workflow languages (graphical and textual) tend to be very procedural, i.e., after each step in the process it is explicitly specified what the next step will be. As a result, the designer tends to over-specify the process leading to all kinds of exceptions and changes. By using more declarative language one tends to under-

specify the process thus providing workers with more space to maneuver.

Each of these approaches can be combined with the four-step approach depicted in Figure 1. However, further work is needed to develop EUCs and CWNs that can capture the degree of required flexibility and link this to concrete workflow languages allowing for more flexibility. Currently, even the most innovative systems support only one form of flexibility. For example, Adept [28] only supports dynamic change, FLOWer [6] only supports case handling, YAWL [7] only supports worklets, and Declare only supports declarative languages [4, 24]. Hence, future work will aim at an analysis of the various forms of flexibility in the context of the approach presented in this paper.

Acknowledgements

We thank Søren Lauesen for permission to use the task descriptions for the Fyn County EPR as basis for this paper. We also thank Søren for fruitful discussions and feedback on this paper.

References

1. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
3. W.M.P. van der Aalst, J.B. Jørgensen, and K.B. Lassen. Let's Go All the Way: From Requirements via Colored Workflow Nets to a BPEL Implementation of a New Bank System. In *Proc. of 13th International Cooperative Information Systems Conf.*, volume 3760 of *LNCS*, pages 22–39, Agia Napa, Cyprus, 2005. Springer.
4. W.M.P. van der Aalst and M. Pesic. DecSerFlow: Towards a Truly Declarative Service Flow Language. In M. Bravetti, M. Nunez, and G. Zavattaro, editors, *International Conference on Web Services and Formal Methods (WS-FM 2006)*, volume 4184 of *Lecture Notes in Computer Science*, pages 1–23. Springer-Verlag, Berlin, 2006.
5. W.M.P. van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT Press, 2002.
6. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
7. M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Facilitating Flexibility and Dynamic Exception Handling in Workflows. In O. Belo, J. Eder, O. Pastor, and J. Falcao e Cunha, editors, *Proceedings of the CAiSE'05 Forum*, pages 45–50. FEUP, Porto, Portugal, 2005.
8. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.
9. C. Bossen and J.B. Jørgensen. Context-descriptive Prototypes and Their Application to Medicine Administration. In *Proc. of Designing Interactive Systems (DIS) 2004*, pages 297–306, Cambridge, Massachusetts, 2004. ACM Press.
10. A. Cockburn. *Writing Effective Use Cases.* Addison-Wesley, 2000.
11. Krav til Fyns Amts EPJ-system (udkast) — Requirements to Fyn County's EPR System (Draft). Fyns Amt, 2003.
12. P. Grünbacher, A. Egyed, and N. Medvidovic. Reconciling software requirements and architectures with intermediate models. *Software and Systems Modeling*, 3(3):235–253, 2004. Springer.
13. D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.
14. M. Jackson. *Problem Frames — Analyzing and Structuring Software Development Problems.* Addison-Wesley, 2001.
15. K. Jensen. *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1992.
16. 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.
17. J.B. Jørgensen and C. Bossen. Executable Use Cases as Links Between Application Domain Requirements and Machine Specifications. In *Proc. of 3rd International Workshop on Scenarios and State Machines (at ICSE 2004)*, pages 8–13, Edinburgh, Scotland, 2004. IEE.
18. J.B. Jørgensen and C. Bossen. Executable Use Cases: Requirements for a Pervasive Health Care System. *IEEE Software*, 21(2):34–41, 2004.
19. S. Lauesen. *Software Requirements — Styles and Techniques.* Addison-Wesley, 2002.
20. S. Lauesen. Task Descriptions as Functional Requirements. *IEEE Software*, 20(2):58–65, 2003.
21. G. Leonardi, S. Panzarasa, S. Quaglini, M. Stefanelli, and W.M.P. van der Aalst. Interacting Agents through a Web-based Health Serviceflow Management System. *Journal of Biomedical Informatics*, 2007.
22. OMG Unified Modeling Language Specification, Version 2.0. Object Management Group (OMG); UML Revision Taskforce, 2006.
23. M. Peleg and et al. Comparing Computer-interpretable Guideline Models: A Case-study Approach. *Journal of the American Medical Informatics Association*, 10(1):52–68, 2003.
24. M. Pesic and W.M.P. van der Aalst. A Declarative Approach for Flexible Business Processes. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Dynamic Process Management*

(DPM 2006), volume 4103 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, Berlin, 2006.

25. S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. Guideline-based Careflow Systems. *Artificial Intelligence in Medicine*, 20(1):5–22, 2000.
26. 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.
27. H.A. Reijers and W.M.P. van der Aalst. The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned. *International Journal of Information Management*, 25(5):458–472, 2005.
28. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.
29. I. Sommerville. *Software Engineering — Seventh Edition.* Addison-Wesley, 2004.
30. T. Wendler, K. Meetz, and J Schmidt. Workflow Automation in Radiology. In H.U. Lemke, editor, *Proceedings of Computer Assisted Radiology and Surgery (CAR98)*, pages 364–369. Elsevier, 1998.
31. Michael Westergaard and Kristian Bisgaard Lassen. Building and Deploying Visualizations of Coloured Petri Net Models Using BRITNeY animation and CPN Tools. In Kurt Jensen, editor, *Procedings of the Sixth CPN Workshop*, volume PB-576 of *DAIMI*, pages 119–136, Arhus, Denmark, October 2005.
32. Michael Westergaard and Kristian Bisgaard Lassen. The BRITNeY Suite Animation Tool. In Susanna Donatelli and P.S. Thiagarajan, editors, *Proceedings of the 27th International Conference on Application and Theory of Petri Nets and Other Models Of Councurrency*, volume 4042, pages 331–340, Turku, Finland, June 2006. Springer-Verlag Berlin Heidelberg 2006.
33. CPN Tools. www.daimi.au.dk/CPNTools.