Executable Use Cases for Pervasive Healthcare

Jens Bæk Jørgensen and Claus Bossen
Centre for Pervasive Computing
Department of Computer Science, University of Aarhus,
IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark
email: jbj@daimi.au.dk, bossen@daimi.au.dk

Abstract

Using a pervasive healthcare system as example, a new approach to specification of user requirements for pervasive IT systems is presented. A formal modelling language, Coloured Petri Nets, is applied to describe what we call Executable Use Cases, EUCs. EUCs are precise, detailed, and executable descriptions of future work processes and their computer support. In particular, EUCs allow user requirements specifications to take the frequently changing context of the users, e.g. their location and equipment in possession, into account.

Topics: Specification of user requirements, use cases, Unified Modeling Language (UML), application of Coloured Petri Nets to pervasive systems, context awareness.

1 Introduction

Worldwide, many hospitals are in the process of introducing *electronic patient records, EPRs*, or have already done so [1]. Though a vision of an EPR already appeared in the late 1960s, IT technology apparently only reached a level that enabled workable solutions in the 1990s, when the spread of EPRs began to gain momentum [6]. In Denmark, Aarhus County has initiated development of an EPR [18] that will substitute several paper-based core documents used for documentation and communication within hospitals today. The aim is to enhance the quality of healthcare, e.g. by allowing multiple users to work on the same, always up-to-date patient record at the same time.

The EPR of Aarhus County, and indeed any EPR, solves obvious problems occurring with paper-based patient records such as being not always up-to-date, mislaid, or even lost. However, EPRs also have their drawbacks and potentially induce at least two central problems for their users. The first problem is *immobility*: in contrast to a paper-based record, an EPR accessed from stationary desktop PCs cannot be easily transported. The second problem is *time-consuming login and navigation*: EPR requires user identification and login to ensure information confidentiality and integrity, and to start using EPR for clinical work, a logged-in user must navigate, e.g. to find a specific document for a specific patient. *Pervasive computing* [2, 7, 8] is a candidate approach to alleviate these two problems. Specifically, a new *pervasive healthcare system* [3, 21]

is being envisioned in a joint project between Aarhus County Hospital, the software company Systematic Software Engineering A/S [23], and the Centre for Pervasive Computing [19] at the University of Aarhus. This paper focuses on specification of the *user requirements* for that system.

One of the dominant approaches to specify user requirements in today's object-oriented system development projects is *use cases* [4, 9, 12] as defined in the Unified Modeling Language, UML [13, 15]. Use cases model work processes to be supported by a new IT system, and a set of use cases is interpreted as user requirements for that system. However, UML use cases have several general and known shortcomings, see e.g. [16] which points out a number of problems under headlines like *use case modelling misses long-range logical dependency* and *use case dependency is non-logical and inconsistent*. For pervasive systems, new complexities are added to the specification of user requirements in order to cope with issues like mobility and context awareness, i.e. the ability of the IT system to react sensibly to various changes of context such as users moving from one location to another.

For these reasons, we will suggest a new notion of use cases. Inspired by the UML use case approach, we will also create models of work processes, but will do so in the formal modelling language *Coloured Petri Nets, CPN* [10, 11, 20]. CPN models are precise, detailed, and, as a particularly valuable asset, executable. Furthermore, as we will show, CPN is suitable to describe *context aware systems*. Contexts can be captured by an elaborated state notion of CPN, and the functionality of a system in a current context can be modelled by an appropriate action notion of CPN. The main contribution of the paper is to introduce and justify the notion of *Executable Use Cases, EUCs*, based on CPN, for the specification of user requirements for pervasive IT systems.

The paper provides a basis introduction to CPN, and, thus, does not assume the reader to be familiar with neither CPN, nor Petri nets [14] in general. The structure is as follows: Section 2 describes the concept of *pervasive healthcare* and the *pervasive medicine administration* work process that will be used as running example. Section 3 is a primer on CPN, and Section 4 presents an EUC for pervasive medicine administration. In Section 5, we report on how to interpret an EUC as actual user requirements. The conclusions are drawn in Section 6.

2 Pervasive Healthcare

The pervasive healthcare system considered in this paper was envisioned in a series of workshops with participation of physicians, nurses, computer scientists, and an anthropologist (one of the authors). Moreover, input came from ethnographic fieldwork by the anthropologist, who spent two months at a department at Aarhus County Hospital in spring 2001. Observation of existing work processes and use of the paper-based patient records confirmed that immobility and time-consuming login and navigation procedures are severe, potential obstacles to the success of EPR.

2.1 Characteristics of Hospital Work Processes

The EPR immobility problem should be solved in order to preserve the inherent mobility of hospital work processes. Paper patient records are very mobile and now frequently moved, e.g. between the nurses' office, the medicine cabinet room, and the wards with patient beds. The other anticipated problem, time-consuming login and navigation, is crucial to overcome in order to allow healthcare personnel to make flexible and smooth transition from one work process to another. This is needed, because interruption and later resumption of work processes are frequent: the personnel continuously have to reschedule their plans for their shift, since new tasks are often added to their list of what to do. New, acutely ill patients are admitted to the department, examinations show a need for immediate action, the condition of a patient deteriorates suddenly, scheduled examinations at other departments are cancelled, etc.

The present plans for EPR deployment entails an additional, severe problem: restricted access to the patient records. The paper-based patient record comprises three separate documents for each patient, a physician's patient record, a nurse's patient record, and a medicine plan. These documents are kept at different places most of the day. Thus, even though only one person can access a specific part of a patient record at a time, there are in the considered department, with 25 patients, nevertheless 75 different access points to the records (three per patient). With EPR, accessibility is proportional with the number of available computers – an obvious bottleneck. The department at present budgets for eight desktop PCs to be placed in offices and two laptops which can be brought along to the wards. In this way, the existing 75 access points decrease to only 8+2=10.

The work process in focus in this paper is *medicine administration*, i.e. handling of medicine for patients. It involves *medicine plans*, which for each patient specify the prescribed medicine, and are used by the nurses to acknowledge when medicine has been poured and given. Medicine plans are usually kept by the medicine cabinet and often taken to the wards together with the medicine. This allows nurses to promptly acknowledge giving of medicine at the wards and to answer questions from patients about their medication. To enable a smooth medicine administration process with EPR, possible solutions with today's technology seem to be computers in each ward or widespread use of personal digital assistants, PDAs – laptops are too heavy to be regularly carried around.

2.2 Pervasive Healthcare System Design Principles

A prototype of the pervasive healthcare system has been created and subsequently tested by healthcare personnel from Aarhus County Hospital [3]. The prototype is built upon three general design principles. The first principle is *context awareness*. This means that the system is able to register and react upon certain changes of context. More specifically, nurses, medicine trays, patients, beds, and other items are equipped with radio frequency identity, RFID, tags [22], such that presence of such items can be detected automatically by involved context aware computers, e.g. located by the medicine cabinet and by the patient beds.

The second design principle is that the system is *propositional*, in the sense that it makes qualified propositions, or guesses. Context changes may result

in automatic generation of buttons, which appear at the taskbar of computers. Users must explicitly accept a proposition by clicking a button – and implicitly ignore or reject it by not clicking. The presence of a nurse holding a medicine tray for patient P in front of the medicine cabinet is a context that triggers automatic generation of a button `Medicine plan: Patient P`, because in many cases, the intention of the nurse is now to navigate to the medicine plan for patient P. If the nurse clicks the button, she is logged in and taken to patient P's medicine plan. It is of course impossible always to guess the intention of a user from a given context, and without the propositional principle, automatic shortcutting could become a nuisance, because of guesses that would sometimes be wrong.

The third design principle is that the system is *non-intrusive*, i.e not interfering with or interrupting hospital work processes in an undesired way. Thus, when a nurse approaches a computer, it should react on her presence in such a way that a second nurse, who may currently be working on the computer, is not disturbed or interrupted. The last two design principles cooperate to ensure satisfaction of a basic mandatory user requirement: important hospital work processes have to be executed as conscious and active acts by responsible human personnel, not automatically by a computer.

2.3 Pervasive Medicine Administration

Work process descriptions in natural language were made as a result of the workshops, and these descriptions formed the basis for the prototype discussed above. The work process constituting the scope of this paper is *pervasive medicine administration*. It covers medicine administration as carried out by personnel supported by the pervasive healthcare system, and is outlined in the following in a style resembling a main constituent of a traditional UML use case.

Assume that nurse N wants to pour medicine into a medicine tray and give it to patient P. First, the nurse goes to the room containing the medicine cabinet. Here is a context aware computer on which the buttons `Login: Nurse N` and `Patient list: Nurse N` appear on the taskbar, when the nurse approaches. Assume that the second button is clicked. Then, N is logged in and a list of those patients for which she is in charge is displayed on the computer.

A medicine tray must be associated with each patient. If a medicine tray is already associated with patient P, the button `Medicine plan: Patient P` will appear on the taskbar of the computer, when the nurse takes the tray nearby, and a click will make P's medicine plan appear on the display. However, if P is a newly admitted patient, it is necessary to associate a medicine tray to him. Nurse N does so by taking an empty tray from a shelf and making the association. In either case, N pours medicine into the tray, acknowledges this in EPR, and is automatically logged out, when she leaves the medicine cabinet area.

Nurse N now takes patient P's medicine tray and goes to the ward where P lies in a bed, which is supplied with a context aware computer. When N approaches, the buttons `Login: Nurse N`, `Patient list: Nurse N`, and `Medicine plan: Patient P` will appear on the taskbar. If the last button is clicked, the medicine plan for P is displayed. Finally, N gives the medicine tray to P, acknowledges the giving in EPR, and is automatically logged out again, when she leaves the bed area.

This rather straight flow of events has numerous variations, e.g. medicine may be poured for one or more patients, for only one round of medicine giving, all four regular rounds of a 24 hours period, or for ad hoc giving; a nurse may have to fetch trays left at the wards prior to pouring; a nurse may approach the medicine cabinet without intending to pour medicine, but only to log into EPR or wanting to check an already filled medicine tray. To enable a smooth pervasive medicine administration work process, the pervasive healthcare system must be specified to handle all these variations and many more. The Executable Use Case, EUC, apparatus is able to do that. We will present an EUC for pervasive medicine administration in Section 4, but first give an informal and general introduction to some fundamental concepts of the EUC modelling language CPN in the following section.

3 EUC Modelling Language – CPN

Coloured Petri Nets, CPN [10, 11, 20], is a mature and well-proven modelling language suitable to describe the behaviour of systems with concurrency, resource sharing, and synchronisation. A CPN model resembles a board game, with strict rules that define the possible executions of the model. The CPN modeller's task is to specify an appropriate board, tokens, and playing rules to reflect the domain being modelled.

We will introduce the basic CPN concepts by means of the simple `Provide trays` model, shown in Figure 1. This model describes how nurses provide medicine trays prior to pouring, checking, and giving medicine.

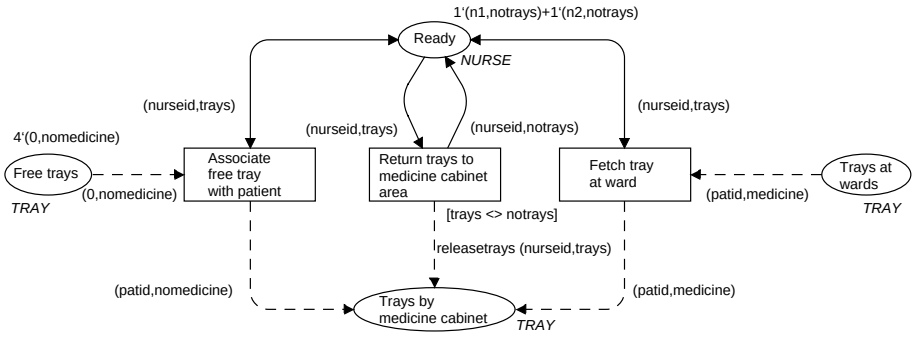

Figure 1: `Provide trays`.

3.1 Modelling of States

A CPN model describes both the states and the actions of a system. States capture the contexts in which actions may take place. The *state* of a CPN model is a distribution of *tokens* on the *places*. Each place is drawn as an ellipse and has an associated *data type*, written in italic capital letters, which determines the kinds ("colours") of tokens the place is allowed to contain.

In Figure 1, a token on the `Ready` place models that a real-life nurse is in a situation where she is ready to carry out work, e.g. she sits at her office.

Thus, `Ready` has the data type `NURSE`, denoting nurses. A nurse is represented as a pair `(nurseid,trays)`, where `nurseid` identifies the nurse and `trays` is a container data structure holding the medicine trays that this nurse currently has in possession.

The places `Trays at wards` and `Trays by medicine cabinet` have data type `TRAY`, and model the trays which currently are at the indicated locations. A tray is modelled as a pair `(patid,medicine)`, where `patid` identifies the patient that the tray is associated with, and `medicine` is a container data structure holding the medicine that currently is in the tray. The place `Free trays` also has data type `TRAY`. It holds tokens corresponding to trays which are not yet associated with patients. The patient id 0 is a special value, used in `TRAY` tokens on the form `(0,nomedicine)`, which represent empty trays, not currently associated with any patient.

The *initial state* describes the start state of the model, before execution begins. In the initial state, there are two `NURSE` tokens on the `Ready` place, `n1` and `n2`, both with no trays, and four empty, non-associated `TRAY` tokens on the `Free trays` place, as indicated by the inscriptions close to the places (an expression like `c1‘e1` denotes a multiset (a bag) containing `c1` appearances of `e1` tokens). The places `Trays at wards` and `Trays by the medicine cabinet` are empty in the initial state.

3.2 Modelling of Actions

The *actions* of a CPN model are represented using *transitions*, drawn as rectangles. Thus, in Figure 1, a nurse who is ready – corresponding to a token in the `Ready` place – may choose to do one of three possible actions, modelled by the three transitions named `Associate free tray with patient`, `Return trays to medicine cabinet area`, and `Fetch tray at ward`.

A transition and a place may be connected by an *arc*. Solid arcs show the flow of `NURSE` tokens, and dashed arcs the flow of `TRAY` tokens (different graphical appearances are used only to enhance readability, and have no formal meaning). The actions of a CPN model consist of transitions removing tokens from input places and adding tokens to output places, often referred to as the *token game*. Input/output relationship between a place and a transition is determined by the direction of the connecting arc. A place may be both input and output, e.g. `Ready` relative to all three transitions – a double arc is a shorthand for one arc in each direction. The tokens removed and added are determined by *arc expressions*, e.g. the expression `(nurseid,trays)` on the arc from the `Ready` place to the `Return trays to medicine cabinet area` transition, where `nurseid` and `trays` are variables that can be assigned data values.

The executability of CPN models comes from the fact that the CPN modelling language has a *formal, operational semantics*. A transition which is ready to remove and add tokens is said to be *enabled*, and requires two kinds of conditions to be fulfilled. The first kind is that appropriate tokens are present on the input places. This means that one condition for enabling of the transition `Return trays to medicine cabinet area` is that the only input place, `Ready`, contains some token matching the expression `(nurseid,trays)`. The second kind of condition comes from the *guard*, which is a boolean expression optionally assigned to a transition, and which must evaluate to true for the

transition to be enabled. `Return trays to medicine cabinet area` has the guard `[trays<>notrays]`. This means that a nurse may return trays to the medicine cabinet area, only when she has some in possession.

An enabled transition may *occur*. The occurrence of `Return trays to medicine cabinet area` models that a nurse takes all the trays she is currently possessing and returns them to the medicine cabinet area, and afterwards is ready again. In the CPN model, this is reflected by a token `(nurseid,trays)` removed from `Ready`, a token `(nurseid,notrays)` added to `Ready`, and a token determined by the expression `releasetrays(nurseid,trays)` added to `Trays by medicine cabinet`.

4 Pervasive Medicine Administration EUC

We have now introduced the necessary CPN concepts, allowing us to present an EUC for pervasive medicine administration. The EUC models the work process from the point of view of a nurse, and a prime focus is to describe how the computers which are by the medicine cabinet and by the patient beds react on context changes and how they support the nurses.

In general, a CPN model consists of a number of modules organised in a hierarchical fashion – Section 3 presented just one single module. The pervasive medicine administration EUC CPN model consists of 11 modules, with a total of 54 places and 29 transitions. We will provide a general overview of the model and supplement it with a detailed explanation of one selected typical module.

4.1 Model Overview

An overview of the model in terms of a hierarchy with a node for each module and arcs showing the relationship between the modules is given in Figure 2. The figure shows how the work process pervasive medicine administration is split into sub work processes. An arc between two nodes indicates that the module of the source node contains a *substitution transition*, whose detailed behaviour is described on the module of the destination node, called the *sub-module*.

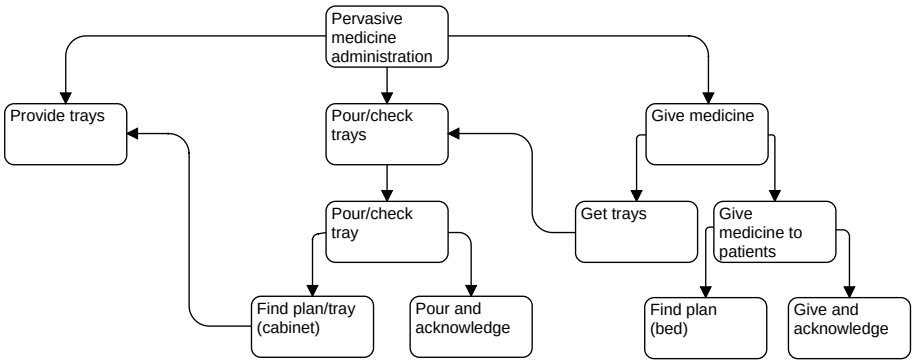

Figure 2: Overview of pervasive medicine administration EUC.

The topmost module `Pervasive medicine administration` of Figure 2 is shown in Figure 3 and contains a very high-level and abstract description of the

work process. All three transitions in Figure 3 are substitution transitions – as indicated by the small `HS` (hierarchy substitution) tag. They correspond to the three arcs emanating from the `Pervasive medicine administration` node in Figure 2, and are briefly described below.

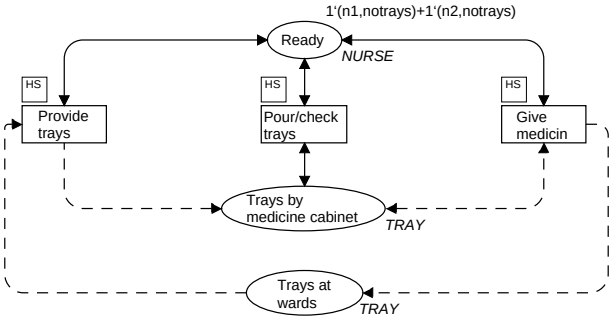

Figure 3: `Pervasive medicine administration` module (topmost node of Figure 2).

4.1.1 Provide trays

We have previously presented and explained this module in Figure 1 in Section 3. From Figure 3, it is possible to see that a nurse sometimes provides trays, by carrying out the action corresponding to the `Provide trays` transition. It is not possible to see how she does it. The sub-module that is bound to the substitution transition `Provide trays`, shown in Figure 1, contains the detailed description of how trays can be provided.

The individual modules of a CPN model interact when the model is executed. In the modules of Figures 3 and 1, places with the same name (e.g. the two `Ready` places) are conceptually glued together, thus allowing exchange of tokens between the modules when the token game is played.

4.1.2 Pour/check trays

As can be seen from Figure 2, the module `Pour/check trays` (note plural 's' in `trays`) uses a sub-module called `Pour/check tray`. `Pour/check trays` models how trays are poured and/or checked for a number of patients. `Pour/check tray` models how a tray is poured and/or checked for one single patient. The module `Pour/check trays` may be seen as similar to a loop statement in a programming language and `Pour/check tray` as the body of the loop.

`Pour/check tray` is itself taking advantage of two sub-modules, one called `Find plan/tray (cabinet)` and one called `Pour and acknowledge`. The first module, `Find plan/tray (cabinet)` models how the nurse gets the medication plan for a given patient presented on the screen of the medicine cabinet computer and/or how she provides the patient's tray – as can be seen on Figure 2, possibly using the `Provide trays` module already described. The `Pour and acknowledge` module models how the nurse actually pours medicine into the tray and how she acknowledges to the pervasive healthcare system when a certain medicine type has been poured.

4.1.3 Give medicine

As can be seen from Figure 2, the module `Give medicine` uses two sub-modules, `Get trays` and `Give medicine to patients`. `Get trays` models how the nurse collects a number of trays (corresponding to a number of patients), before she embarks on a round to the wards to give medicine. This preparation process may involve checking and additional pouring of existing partly or fully filled trays, and therefore `Get trays` uses the `Pour/check trays` module. The `Give medicine to patients` and its two sub-modules `Find plan (bed)` and `Give and acknowledge` are similar to the `Pour/check tray (cabinet)` and its two sub-modules.

4.2 Module Example – `Pour/check trays`

We now present and explain a typical module from the model, the `Pour/check trays` module, shown in Figure 4.

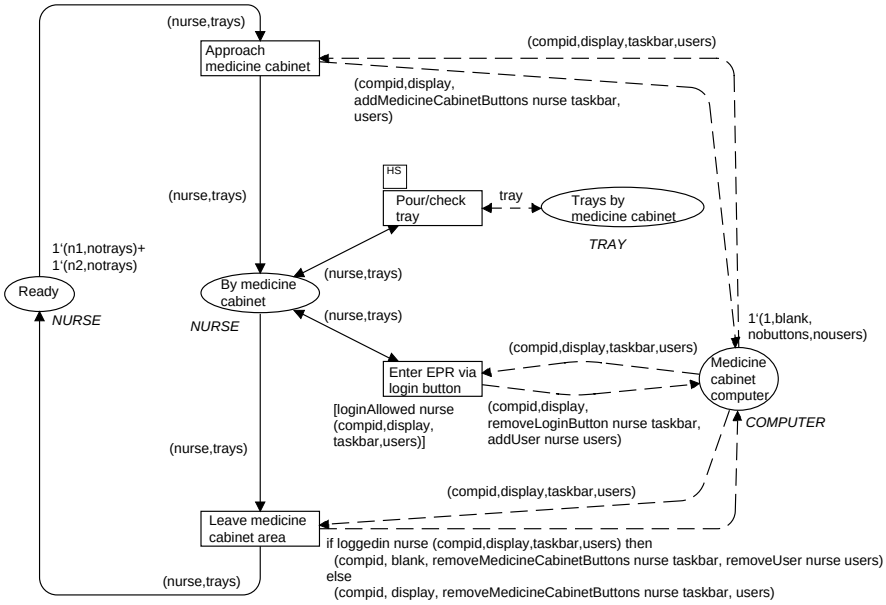

Figure 4: `Pour/check trays` module.

On this module, the medicine cabinet computer is in focus. The computer is modelled by a token on the `Medicine cabinet computer` place having data type `COMPUTER` – a 4-tuple `(compid,display,taskbar,users)` consisting of a computer identification, its display (main screen), its taskbar, and its current users. Long-dashed arcs show the flow of the `COMPUTER` token. Recall that the medicine cabinet computer is context aware, i.e. it is able to sense the proximity of both nurses and trays.

In the initial state, two nurses `n1` and `n2` are ready and have no trays, corresponding to two tokens in the `Ready` place. The medicine cabinet computer is idle, with a blank display, no taskbar buttons, and no current users.

Occurrence of the `Approach medicine cabinet` transition models that a nurse changes from being ready to being busy nearby the medicine cabinet. Moreover, at the same time, two buttons are added to the taskbar of the medicine cabinet computer, namely one login button for the nurse and one patient list button for the nurse. In the CPN model, these taskbar buttons are added by the function `addMedicineCabinetButtons` appearing on the arc going from the transition `Approach medicine cabinet` to the place `Medicine cabinet computer`.

The possible actions for a nurse who is by the medicine cabinet are modelled by the three transitions `Pour/check tray`, `Enter EPR via login button`, and `Leave medicine cabinet area`. Often, a nurse by the medicine cabinet wants to pour and/or check some trays. How this pouring and checking is carried out is modelled on the sub-module `Pour/check tray`, which is bound to the substitution transition having the same name (as can be seen from Figure 2).

The `Enter EPR via login button` transition models that a nurse clicks on the login button and makes a general-purpose login to EPR. It is outside the scope of the model to describe what the nurse subsequently does – the domain of the model is specifically pervasive medicine administration, not general EPR use. The transition has a guard which ensures that only a user who is not currently logged into EPR can do so. When a nurse logs in, the login button for that nurse is removed from the taskbar of the computer, modelled by the `removeLoginButton` function on the arc from `Enter EPR via login button` to the `Medicine cabinet computer` place. Moreover, the nurse is added to the set of current users by the function `addUser` appearing on the same arc.

The `Leave medicine cabinet area` transition models that a nurse has done the pouring and checking that she wants for now, and leaves the area. Upon leaving, it is checked whether the nurse is currently logged in, modelled by the function `loggedIn` appearing in the if-then-else expression on the arc going from `Leave medicine cabinet area` to the `Medicine cabinet computer` place. As can be seen further from the expression on that arc, if the nurse is logged in, the medicine cabinet computer automatically blanks off the screen, removes her taskbar buttons (`removeMedicineCabinetButtons`), and logs her off (`removeUser`). If she is not logged in, the buttons generated because of her presence are removed, but the state of the computer is otherwise left unaltered. When a nurse leaves the medicine cabinet area, the corresponding token is put back on the `Ready` place.

5 EUCs as User Requirements

From the EUC for pervasive medicine administration, a list of user requirements can be produced by focusing on the model transitions which manipulate the involved computers. Each transition connected to the places `Medicine cabinet computer` (shown) and `Bed computers` (not shown) must be taken into account. As examples, the requirements below are induced by the transitions on the module `Pour/check trays` shown in Figure 4:

- When a nurse approaches the medicine cabinet area, the medicine cabinet computer must add a login button and a patient list button for that nurse to the taskbar.
- When a nurse leaves the medicine cabinet area, if she is logged in, the

medicine cabinet computer must blank off its display, remove the nurse's login button and patient list button from the taskbar, and log her off.

- When a nurse logs into EPR, her login button must be removed from the taskbar of the computer, thus disallowing the nurse to log in again, if she already is.

The first two of these three user requirements can also partly be derived from the prose English description of pervasive medicine administration in Section 2.3, i.e. these two requirements were known after the workshops, prior to creation of the EUC. However, the EUC states the requirements more precisely and with more details, e.g. the EUC specifies that the computer display should be blanked when a logged in nurse leaves the medicine cabinet area, and it also describes what should happen when a nurse, who is not logged in, leaves.

In general, a user requirements specification should be precise and address as many issues as possible. We will argue that to a higher extent than reading a static prose descriptions of future work processes as the one in Section 2.3, application of EUCs forces many questions to be asked. An EUC is a dynamic, executable model, allowing the behaviour of a future work process to be visualised and investigated. Playing the token game of an EUC typically catalyses the participants' cognition and generates new ideas. In this way, it often happens that questions that the participants had not thought about earlier appear. Examples of questions (Qs) that have appeared during execution of the EUC of Section 4, and corresponding answers (As), are:

- Q: what happens if two nurses both are close to the medicine cabinet computer? A: the computer generates login buttons and patient list buttons for both of them.

- Q: what happens when a nurse carrying a number of medicine trays approaches a bed? A: in addition to a login button and a patient list button for that nurse, only one medicine plan button is generated – a button for the patient associated with that bed.

- Q: is it possible for one nurse to acknowledge pouring of medicine for a given patient while another nurse at the same time acknowledges giving of medicine for that same patient? A: no. That would require a systems architecture allowing a higher degree of concurrency and a more fine-grained concurrency control exercised over the patient records.

Questions like the ones above may imply changes to be made to the EUC, because emergence of a question indicates that the current version of the EUC does not reflect the future work process properly. As a concrete example, in an early version of the pervasive medicine administration EUC, the exit of any nurse from the medicine cabinet area resulted in the computer screen being blanked off. To be compliant with the non-intrusive design principle, the exit of a nurse who is not logged in, should of course not disturb another nurse who might be working at the computer, and the EUC had to be changed accordingly.

Specification of user requirements is a prominent, general problem in the software industry today. In many development projects, the user requirements are initially too vaguely specified and too poorly understood. This is quite

unfortunate, because a user requirements specification for a software system is often an essential part of a legal contract between a customer, e.g. a hospital, and a software company. Questions like the three above may easily be subject to dispute. However, if the parties have agreed that pervasive medicine administration should be supported, and have the overall stipulation that the EUC presented in the previous section is the authoritative description, many disagreements can quickly be settled, because of the formality and unambiguity of the EUC.

EUCs are not a panacea. Their purpose is solely to describe the user requirements of a future pervasive IT system, relative to the work flow to be supported. A number of other user requirements issues regarding IT systems support of pervasive medicine administration cannot be addressed properly by EUCs, e.g.:

- *User interface*, e.g. where are the buttons placed, what do they look like, and how is a medication plan presented on the computer screen?
- *Response times*, e.g. how long should a nurse wait before her buttons appear on the screen?
- *Distance*, e.g. how close should a nurse be to a computer, before it is aware of her presence?

6 Conclusions

The pervasive healthcare system, and many pervasive systems in general, are characterised by classical and well known complications that apply to many distributed systems [5], plus a number of new problems to be tackled, e.g. regarding context awareness and mobility. With more complex systems come increased demands to the modelling languages that we use for their specification and development. In this paper, the notion of Executable Use Cases, EUCs, based CPN, is introduced. CPN is one dialect of Petri nets [14], and various kinds of Petri nets have previously been used for modelling of work flows [17], much in the same way as in the EUC approach. However, application of Petri nets to specification of user requirements for pervasive systems is, to the best of our knowledge, new.

UML is a de facto standard for object-oriented modelling in the software industry, and UML use cases are very popular for specification of the user requirements for many of today's IT systems. We believe that UML use cases are not always sufficient for pervasive systems. A viable alternative is EUCs, which have three main strengths compared to UML use cases in general: *precision*, *detail*, and *executability*, cf. the comparison in Section 5 of the UML use case style prose description of pervasive medicine administration of Section 2.3 and the EUC of Sect. 4. An additional strength of EUCs and CPN is the rich, elaborated, and precise notion of state, which is not commonly found in other modelling languages, and certainly not in UML. With the state notion, the modelling of contexts comes very natural, e.g. the frequently occurring context element location is conveniently captured by means of CPN places, cf. place names like `By medicine cabinet` in the EUC of this paper. This makes EUCs suitable to specify user requirements for pervasive, context aware systems.

As cited in the introduction, one of the main critiques raised against UML use cases is that they promote a highly localised perspective [16], and do not properly capture dependencies between various sub work processes. For the pervasive medicine administration work process, this general problem has materialised as being difficult and cumbersome to describe with prose text in a sufficiently precise fashion the many dependencies between the involved sub work processes. Such dependencies are explicitly and precisely captured in the EUC, cf. the EUC overview in Figure 2.

EUCs, of course, have potential drawbacks. One drawback is that CPN is indeed a formal language (in other situations, an advantage, though), and thus difficult to use as a means of communication between system developers and non-technical end users. In contrast, one of the often cited strengths of UML use cases is that they are non-formal and easy to understand for users. Like UML use cases, for the best result, EUCs should be worked out with participation of the future system users, in an iterative fashion going from a coarse and probably not entirely correct representation of a future work process, to more and more mature versions of the EUC, where precision and detail are added in each iteration. To ease communication with healthcare personnel, we have started a project to make the EUC CPN model of this paper the logical controller of a small movie-like computer animation displaying work processes at a hospital department with nurses, medicine trays, medicine cabinets, wards, beds, computers, etc. In this way, instead of looking directly at the token game of the CPN model, the nurses and physicians will see their future work situation illustrated in a fashion which is much more natural for them, and where the motions of pictures are controlled by, and thus guaranteed to be consistent with, the execution of the CPN model.

Another drawback of EUCs is that CPN is often thought of as being rather complex and time-consuming to learn and to apply. It is true that it takes an effort to learn. However, once learned, it is quite efficient to apply. The EUC of this paper was created by the authors of this paper using a total effort of approximately 60 man hours, 50 hours used by a computer scientist, who is an experienced CPN modeller, and 10 hours used by the anthropologist who has a detailed knowledge of the work processes at the hospitals. The EUC was created in four iterations where the computer scientist was the active modeller and the anthropologist a participant in executions of the various versions of the EUC, and the main source for questions that provided input to the next, more complete, more detailed, and more precise version of the EUC.

Compared to UML use cases, EUCs do potentially require an extra effort during specification of user requirements. However, we think that in many cases, the investment is well justified. Even though many contracts recognise that the user requirements may change during a project (incurring a price adjustment), it is wiser, cheaper, less frustrating, and more efficient for all parties to properly address as many issues as possible already in the initial specification.

References

[1] M. Berg. Accumulating and Cordinating: Occasions for Information Technologies in Medical Work. In *Computer Supported Cooperative Work*, volume 8, 1999.

[2] J. Burkhardt, H. Henn, S. Hepper, K. Rintdorff, and T. Schäck. *Pervasive Computing – Technology and Architecture of Mobile Internet Applications*. Addison-Wesley, 2002.

[3] H.B. Christensen, J. Bardram, and S. Dittmer. Theme One: Administration and Documentation of Medicine. Report and Evaluation. TR-3. Technical report, Centre for Pervasive Computing, Aarhus, Denmark, 2001.

[4] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000.

[5] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems – Concepts and Design*. Addison-Wesley, 2001.

[6] R.S. Dick, E.B. Steen, and D.E. Detmer. *The Computer-Based Patient Record: An Essential Technology for Health Care*. National Academy Press, 1997.

[7] U. Hansmann, L. Merk, M.S. Nicklous, and T. Stober. *Pervasive Computing Handbook*. Springer Verlag, 2001.

[8] A. Helal, B. Haskell, J.L. Carter, R. Brice, D. Woelk, and M. Rusinkiewicz. *Any Time, Anywhere Computing – Mobile Computing Concepts and Technology*. Kluwer Academic Publishers, 1999.

[9] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[10] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Volume 1-3*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 1992-97.

[11] L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.

[12] D. Kulak, E. Guiney, and E. Lavkulich. *Use Cases: Requirements in Context*. Addison-Wesley, 2000.

[13] OMG Unified Modeling Language Specification, Version 1.4. Object Management Group (OMG); UML Revision Taskforce, 2001.

[14] W. Reisig. *Petri Nets, an Introduction*. Springer-Verlag, 1985.

[15] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[16] A.J.H. Simons and I. Graham. 30 Things That Go Wrong in Object Modelling with UML 1.3. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*. Kluwer Academic Publishers, 1999.

[17] W.M.P. van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[18] Aarhus Amt Electronic Patient Record. www.epj.aaa.dk.

[19] Centre for Pervasive Computing. www.pervasive.dk.

[20] Coloured Petri Nets at the University of Aarhus. www.daimi.au.dk/CPnets.

[21] Pervasive Healthcare. www.healthcare.pervasive.dk.

[22] Radio Frequency Identification. www.rfid.org.

[23] Systematic Software Engineering A/S. www.systematic.dk.