Implementation of a healthcare process in four different workflow systems

Implementation of a Healthcare Process in Four Different Workflow Systems

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

[1] Department of Information Systems, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{r.s.mans,w.m.p.v.d.aalst,n.c.russell}@tue.nl

[2] Academic Medical Center, University of Amsterdam, Department of Quality Assurance and Process Innovation, Amsterdam, The Netherlands.
p.j.bakker@amc.uva.nl

Abstract. Currently, many hospitals are investigating the use of a workflow management system in order to provide support for care processes. However, today's workflow management systems fall short in supporting care processes as flexibility is required for its execution. In this paper, we investigate the flexibility requirements that need to be satisfied in order to support healthcare processes having various characteristics. An evaluation shows that different systems need to be used in conjunction with each other in order to fully support the various types of care processes.

1 Introduction

In a competitive health-care market, hospitals have to focus on ways of streamlining their processes in order to deliver high quality and safe care while at the same time reducing costs [10]. Moreover, increasing pressure is being put on hospitals to work in the most efficient way possible given projected increases in demand for hospital care. Consequently, there is a need for technological support in controlling and monitoring healthcare processes for patients [24] and workflow technology is potentially a means for achieving this end. Workflow Management Systems (WfMSs) support processes by managing the flow of work such that individual work items are done at the right time by the proper person [2]

There are a number of benefits for utilizing workflow technology. Processes supported by workflow systems can be executed faster and more efficiently [22]. In addition, these processes can be monitored, and consequently can be executed more rapidly which also leads to increased patient safety.

A number of difficulties commonly arise when hospitals attempt to automate healthcare processes as a consequence of the fact that these processes are *diverse*, and require *flexibility* and that *several medical departments* can be involved in the diagnostic and treatment process. For a group of patients with the same diagnosis, the number of different examinations and treatments required can be high and the order in which they are conducted can vary greatly. Due to intermediary results of diagnostic examinations, the way a patient reacts to the treatment provided, and the condition of the patient themself, it may be

necessary to adapt the care process for a particular patient [12]. Although care processes tend to be ad-hoc and dynamic, it is felt that in the future more *standardization* is needed. This is illustrated by the growing use and acceptance of evidence based protocols and guidelines.

Therefore, an interesting and challenging question arises: *What are the considerations with regard to process flexibility when applying workflow technology in hospitals?*. Or alternatively, what kinds of flexibility are needed in order to support care processes? To answer this question, we implemented a *representative* healthcare process in four workflow systems. Based on the use of a taxonomy of flexibility, which identifies a range of approaches for achieving process flexibility, we will discuss what kind of flexibility is actually needed in order to support the representative healthcare process and healthcare processes in general. By using the taxonomy it is possible to compare the workflow systems with each other and to identify whether the systems can be applied in the healthcare domain.

As the representative care process, we have taken the diagnostic process of patients visiting the gynecological oncology outpatient clinic in the AMC hospital, a large academic hospital in the Netherlands. The healthcare process under consideration is a large process consisting of around 325 activities. We choose to implement the care process in workflow systems which demonstrate various kinds of flexibility. For this purpose we selected YAWL [3], FLOWer [11], ADEPT1 [35] and Declare [32]. YAWL was chosen because it is a powerful open-source system supplying most of the workflow patterns [27]. FLOWer is considered to be the most successful commercial system providing flexibility. ADEPT1 and Declare are two academic systems providing new and powerful ways of supporting "extreme" flexibility.

This paper is structured as follows: Section 2 introduces the taxonomy of flexibility. Section 3 introduces the gynecological oncology healthcare process in general and a subpart of it in detail. Then, in Section 4, the corresponding implementation in each of the different workflow systems is discussed. Section 5 examines the flexibility support in each of the workflow systems. Related work is outlined in Section 6. Section 7 concludes the paper.

2 Flexibility in Workflow Systems

In this section, we present a range of approaches for achieving process flexibility for the *control flow perspective*. The flexibility of a process is its ability to deal with both foreseen and unforeseen changes by varying those parts of the model which are affected by them, whilst retaining the essential format of those parts that are not impacted by the variations. The various approaches for achieving process flexibility have been documented in the form of a taxonomy. In this paper, we will only discuss a shortened version of it although we will still illustrate each flexibility type. The complete version can be found in [28]. In Section 2.1, each of the flexibility types will be discussed further. Afterwards, in Section 5, we use the taxonomy to evaluate the process flexibility support in four workflow management systems, studied in this paper.

2.1 Flexibility Types in Detail

Each flexibility type provides a means for business processes to respond to changes in their operating environment without necessitating the complete redesign of the underlying process specification, however they differ in the timing and manner in which they operate. First of all, processes can be defined in either an imperative or a declarative way.

- An *imperative approach* focuses on the *precise* definition of how a given set of tasks need to be performed. Typically, it describes constraints on the execution are defined either via links (or connectors) between tasks and/or data conditions associated with them.
- A *declarative approach* focuses on *what* should be done instead of *how*. Constraints, which are defined as relations between tasks, are used to restrict possible task execution sequences.

Note that for the declarative approach, fewer execution paths become possible as more constraints are defined for the process, i.e. constraints limit flexibility. In contrast, to increase flexibility in an imperative process, more execution paths have to be added by extending the model.

In the remainder of this section, each flexibility type is discussed in terms of the following characteristics: *motivation* – the rationale for the flexibility type; *definition* – a concise description of the flexibility type; *scope* – the situations and domains to which the flexibility type applies; and *employment considerations* – an overview of how the flexibility type is put into use.

Flexibility by Design

Motivation When a process operates in a dynamic environment it is desirable to incorporate support for the various execution alternatives that may arise within the process model. At runtime, the most appropriate execution path can be selected from those encoded in the design time process model.

Definition *Flexibility by Design* is the ability to incorporate alternative execution paths within a process model at design time allowing the selection of the most appropriate execution path to be made at runtime for each process instance.

Scope Flexibility by design applies to any process which may have more than one distinct execution trace.

Employment considerations The model is complete and fixed which means that it is completely deterministic and not subject to varying interpretation. All allowed execution paths are encoded in the model.

Determining all possible execution paths in a process definition completely at design-time may be either undesirable from the standpoint of model complexity or impossible due to unknown or unlimited number of possible execution paths. The following three flexibility types provide alternative mechanisms for process flexibility.

Flexibility by Deviation

Motivation Some process instances need to temporarily deviate from the execution sequence described by the associated process model in order to accommodate changes in the operating environment encountered at runtime. For example, it may be appropriate to swap the ordering of the *register patient* and *perform triage* tasks in an emergency situation. The overall process model and its constituent tasks remain unchanged.

Definition *Flexibility by deviation* is the ability for a process instance to deviate at runtime from the execution path prescribed by the original process without altering its process model. The deviation can only encompass changes to the execution sequence of tasks in the process for a specific process instance, it does not allow for changes in the process model or the tasks that it comprises.

Scope The concept of deviation is particularly suited to the specification of process models which are intended to guide possible sequences of execution rather than restrict the options that are available (i.e., they are descriptive rather than prescriptive). These specifications contain the preferred execution of the process, but other scenarios are also possible.

Employment considerations Similar as to flexibility by design the model is complete. However, the actual execution at runtime may vary from the strict sequence implied by the process model, allowing for implicit scenarios. The actual execution varies in the sense that tasks may be skipped, undone, redone or another not yet enabled task may be invoked. Nonetheless, the execution of valid execution paths is guaranteed. A valid execution path for an imperative language is considered to be a directed path, according to the process definition, on which all tasks are either skipped or completed. A valid execution path for a declarative language is a path where no *mandatory* constraints are violated and where, if needed, *optional* constraints are violated.

Flexibility by Underspecification

Motivation When specifying a process model it might be foreseen that in the future, during run-time execution, more execution paths are needed which must be incorporated within the existing process model. Furthermore, often only during the execution of a process instance does it become clear what needs to be done at a specific point in the process. When all execution paths cannot be defined in advance, it is useful to be able to execute an incomplete process model and dynamically add process fragments expressing missing scenarios to it.

Definition *Flexibility by underspecification* is the ability to execute an incomplete process model at run-time, i.e., one which does not contain sufficient information to allow it to be executed to completion. Note that this type of flexibility does not require the model to be changed at run-time, instead the model needs to be completed by providing a concrete realization for the undefined parts.

Scope The concept of underspecification is mostly suitable for processes where it is clearly known in advance that the process model will have to be adjusted at *specific points*, although the exact content at this point is not yet known (and

may not be known until the time that an instance of the process is executed). This approach to process design and enactment is particularly useful where distinct parts of an overall process are designed and controlled by different work groups, but the overall structure of the process is fixed. In this situation, it allows each of them to retain some degree of autonomy in regard to the tasks that are actually executed at runtime in their respective parts of the process, whilst still complying with the overall process model.

Employment considerations At design time, the model is incomplete as there exist tasks in the model which are not fully specified. The ultimate realization of these tasks can be deferred until runtime. However, the execution of valid execution paths is guaranteed. A valid execution path is considered to be a directed path, according to the process definition, on which all underspecified tasks are fully specified (i.e. in an deterministic way).

Flexibility by Change

Motivation In some cases, events may occur during process execution that were not foreseen during process design. Sometimes these events cannot be addressed by temporary deviations from the existing process model, but require the addition or removal of tasks or links from the process model on a permanent basis. This may necessitate changes to the process model for one or several process in- stances; or where the extent of the change is more significant, it may be necessary to change the process model for all currently executing instances.

Definition *Flexibility by Change* is the ability to modify a process model at runtime such that one or all of the currently executing process instances are migrated to a new process model. Unlike the previously mentioned flexibility types the model constructed at design time is modified and one or more instances need to be transferred from the old to the new model.

Scope Flexibility by change allows processes to adapt to changes that are identified in the operating environment. Changes may be introduced both at the level of the process instance and also at that of the process model (also known as change at instance level, and type or model level respectively).

Employment considerations The model is complete but not fixed. This means that the model can be changed for running instances by e.g. adding, removing or relinking tasks. An important criterion is that after the change the model is complete again and that model correctness is guaranteed which has as a consequence that only valid execution paths can be followed.

2.2 Evaluation of Contemporary Offerings

To validate the flexibility types, we investigated the degree of support for them in different workflow management systems [28], specifically ADEPT1 [36], YAWL1 (version 8.2b including the worklet service) [3, 9], FLOWer (version 3.0) [7] and DECLARE (version 1.0) [32, 31]. The selection of these workflow systems was

based on the criterion of supporting process flexibility, which excluded most systems. In fact, most commercial systems provide little support for flexibility. The selected systems cover distinct areas of the workflow system spectrum, such as adaptive workflow (ADEPT1), case handling (FLOWer) and declarative workflow (DECLARE). The evaluation results are summarized in Figure 2, which shows whether the workflow system supports (+) or does not support (-) the respective flexibility type. A detailed evaluation can be found in [28]. None of the evaluated systems provides the full range of flexibility alternatives. Flexibility by design is (to some degree) supported by all offerings. YAWL, which offers the so called worklet service, excels in flexibility by underspecification, ADEPT1 in flexibility by change, FLOWer in flexibility by deviation and DECLARE excels in two areas, namely deviation and change.

Flexibility by	ADEPT1	YAWL	FLOWer	DECLARE
design	+	+	+	+
deviation	–	–	+	+
underspecification	–	+	–	–
change	+	+	–	+

Table 1. Product evaluations

3 Case of gynecological oncology

In this section, we will introduce the diagnostic part of the gynecological oncology healthcare process, which we studied. In Figure 1, a snippet showing the most important part of the process is given. Moreover, for the "referral patient and preparations for first visit" node a part of the corresponding subprocess is shown in Figure 2.

Figures 1 and 2 model the gynecological oncology process using so-called *Colored Workflow Nets (CWN)* [6], and which is a specific class of *Colored Petri Nets (CPNs)* [19, 21]. Furthermore, a CWN is a *workflow model* in which we restrict ourselves to concepts and entities which are common in most workflow languages. To this end, a CWN covers the *control-flow*, *organizational*, *data* and *operation* perspectives. Moreover, a CWN abstracts from implementation details and language/application specific issues. More details about a CWN can be found in [6].

In Figure 1, the topmost page of the CWN model is shown which gives a general overview of the diagnostic process of the gynecological oncology healthcare process in the AMC hospital. We are dealing with a large healthcare process involving 325 different activities. Therefore, it is only possible to show a small part of the *overall* model.

As can be seen in Figure 1, the gynecological oncology process consists of two different processes. The *first* process, which is modeled in the lower part

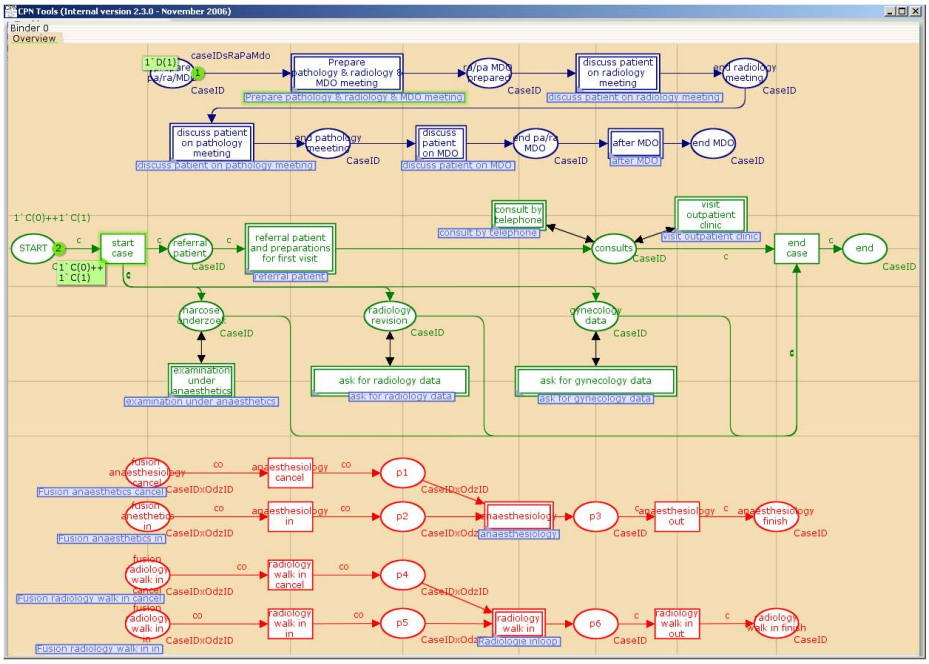

Fig. 1. General overview of the diagnostic process of the gynecological oncology healthcare process. The green and blue nodes and arcs represent respectively the first and second part of the process. The red nodes and arcs represent the interactions with different medical departments.

of the picture and colored green, deals with the diagnostic process that is followed by a patient when referred to the AMC hospital for treatment, up to the point where the patient is diagnosed. In this process, the patient can have several consultations with a doctor, either via visiting the outpatient clinic or via telephone.

During such a consultation, the status of the patient is discussed and a decision is made about whether examinations and consultations need to be requested, canceled or rescheduled. Moreover, during the course of the process, several administrative activities like giving brochures, and registering a patient need to be done, generally by a nurse.

Actually, the doctor can request various tests, performed by different medical departments. The interactions with these medical departments and also the processes adopted by them are modeled at the bottom of Figure 1 and also colored red. More specifically, the interactions with these medical departments are considered as being black boxes where a request for or a cancelation of a test is delivered and deemed to have ended when a result is known or the test is canceled.

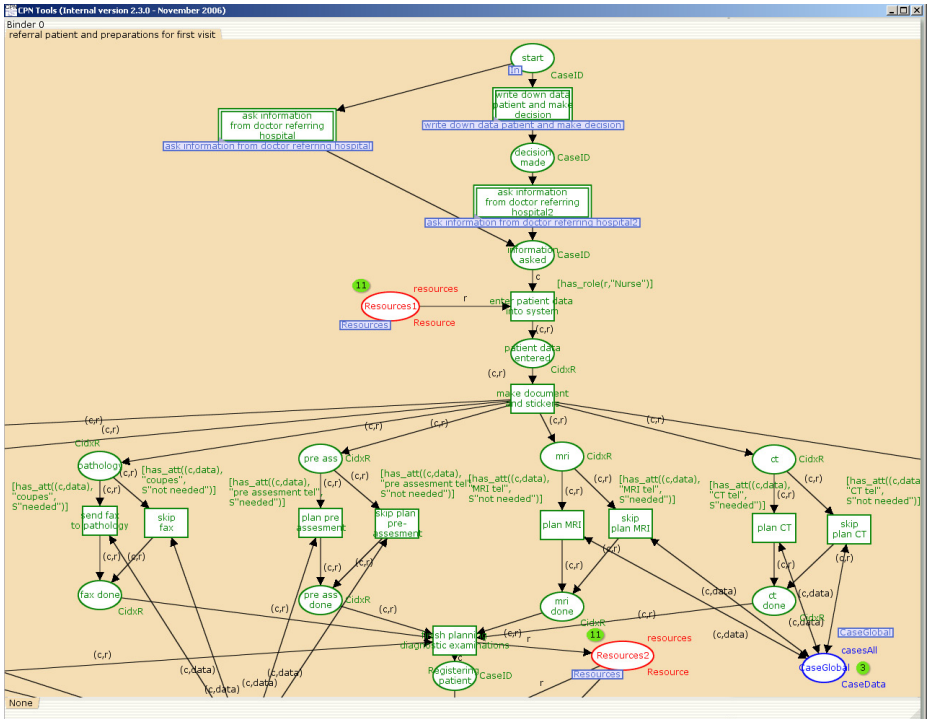

Fig. 2. CWN for the first part of the gynecological oncology healthcare process.

However, it is important to note, that in the future new tests can become available. There can even be new types of medical departments. As a consequence, in Figure 1, it may be necessary to add interactions with new departments. In this way, it becomes clear, that in order to cater for varying interaction with medical departments, support for a wide range of flexibility options is needed in the process. Moreover, as a lot of different tests can be ordered, this has as consequence that a lot of independent processes can run concurrently for a patient. So, while being in the process of visiting the outpatient clinic there may be a series of associated subprocesses running concurrently such that there is a subprocess instance for each test that has been requested and for which still no result is known.

The *second* process, which is modeled completely at the top of the picture and colored blue, deals with the weekly organized meetings, on Monday afternoon, for discussing the status of patients and what needs to be next. There are three meetings involving the departments of radiology, pathology and a multidisciplinary meeting involving the departments of gynecological oncology and radiotherapy.

Now, having introduced the gynecological oncology process, we will focus on its initial stages (substitution transition "referral patient and preparation for

first visit"), in which a doctor of a referring hospital calls a nurse or doctor of the AMC hospital and after which an appointment is made for the first visit of the patient. At that moment it is also decided to additionally schedule appointment for diagnostic tests. This part of the process is shown in Figure 2. For example, we see that the first visit of the patient needs to be planned, and that is possible to make appointments for an "MRI", "CT" or "pre-assessment".

It is important to note that the process, shown in Figure 2, is considered to be a 'standardized procedure' for these patients in the AMC. From the figure, it can be seen that there a number of possible courses of action that may be taken (and the figure only shows halve of the process). Furthermore, as healthcare processes are unpredictable, there can also be the need to *skip* or to *add* an activity. For example, it may be necessary to skip the "plan CT" activity, even though at the beginning of the process the decision has been made to perform a CT scan.

It is impossible to describe the full process and the CPN model as there are more then 300 activities involved resulting in a simulation model of more than 800 nodes. Using interactive simulation and animations, the model was validated by domain experts.

4 Realization of the system in different Workflow Systems

In this section, we will discuss how several different workflow systems have been configured in order to support the healthcare processes. The workflow systems YAWL, FLOWer, ADEPT1 and Declare have been chosen as candidate systems. Each of them demonstrates a specific kind of flexibility, which is deemed relevant when implementing a healthcare process in a workflow system.

In the remainder of this section, we will examine how the flexibility provided by each workflow system has been used or can be utilized during the execution of healthcare processes. Subsequently, in Section 5, we will classify the workflow systems according to the taxonomy presented in Section 2 and then compare them with each other.

4.1 YAWL / Worklets

YAWL (Yet Another Workflow Language) [3] is an open source workflow management system[3], which is based on the well-known workflow patterns[4] [4] and is more expressive than any workflow language available today.

YAWL supports the modeling, analysis and enactment of flexible processes through the use of *worklets* [8] and which can be seen as a kind of configurable process fragment. Specific activities in a process are linked to a repertoire of

[3] YAWL can freely be downloaded from www.yawl-system.com

[4] More information about the workflow patterns can be found on www.workflowpatterns.com

possible actions. Based on the properties of the case and other context information, the desired action is chosen. The selection process is based on a set of rules. Also, during enactment it is possible to add new actions to the repertoire.

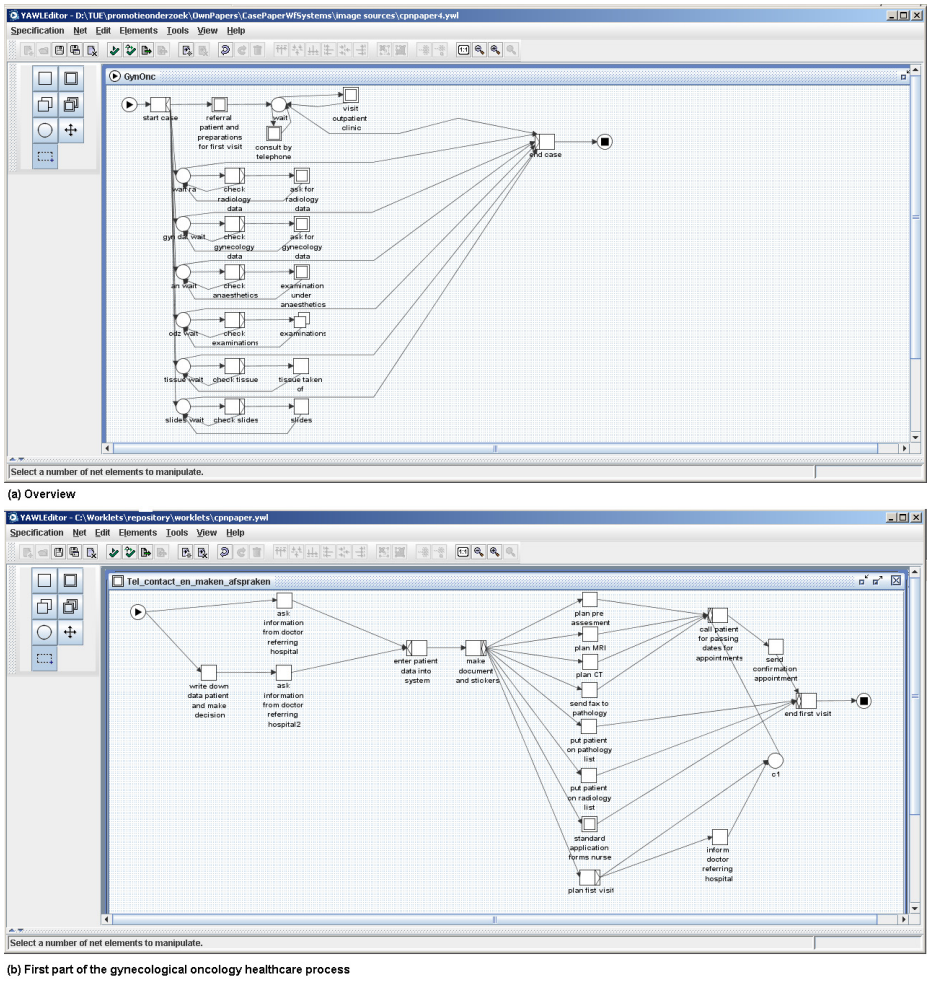

Fig. 3. Screenshots of models in the YAWL editor.

In YAWL, we used the worklet approach for modeling the interactions with all medical departments by linking a "multiple atomic task" node to the worklet service. This is represented in Figure 3(a) by the node with name "examinations" which can be executed multiple times if multiple examinations are needed. In this way, for each test the right worklet can be chosen. In case of a new test, it

is possible to choose a corresponding process fragment, or to dynamically define a new process fragment, and thereby extending the ruleset.

In Figure 3(b), we see the corresponding YAWL process fragment for the first part of the gynecological oncology healthcare process. Due to syntactical sugaring, not the same amount of nodes was needed as had been the case in the CWN model of Figure 2. For example, the "make document and stickers" activity in YAWL is an OR-split, which means that one or more of the outgoing paths may be followed and others may be skipped. This OR-split is used because the "plan MRI" and "plan CT" activities may be either performed or not. By using YAWL's OR-join construct the skip activities which appear in the CWN are not needed in the YAWL model.

4.2 FLOWer

FLOWer is a commercial workflow management system provided by Pallas Athena, the Netherlands[5]. FLOWer is a case-handling product [7]. Case-handling aids process flexibility by focussing on the data aspect rather than on the control-flow aspect. Case-handling offers four core features [7]. The first one is that all information within a case is available, which avoids "context tunneling". Second, the decision of which activities are enabled is based on the information which is available within the case, instead of the activities which have already executed. Third, work distribution is separated from authorization. This allows for additional types of roles, like skipping or redoing activities in the process. In this way, many more (implicit) scenarios are possible within the process. Moreover, a fourth distinguishing feature of FLOWer is that workers are allowed to view and add/modify data before or after the corresponding activities have been executed.

In Figure 4(a), an overview of the FLOWer model is given. In this model, the "examinations" node refers to the interactions with the different medical departments. This node is a so-called "dynamic subplan" which allows for concurrent execution of a sub process. By using guards in this subprocess, the right process will be executed for each test that has been requested. However, in case of a new test it is necessary to change the process definition thus resulting in a new version of the process. As indicated in [29], it can be very risky to update cases which are handled according to the old process definition due to possible process instance deadlock. Consequently, only new cases can be executed according to the new process definition.

In Figure 4(b), we see the FLOWer model for the first part of the gynecological oncology process. When executing this part of the process, it is possible to skip or redo activities. For example, we can skip activity "make document and stickers" and continue with the remainder of the process. Furthermore, if we have already executed activity "confirmation appointment" we still can go back in the process to where we had to execute activity "make document and stickers".

[5] http://www.pallas-athena.com/

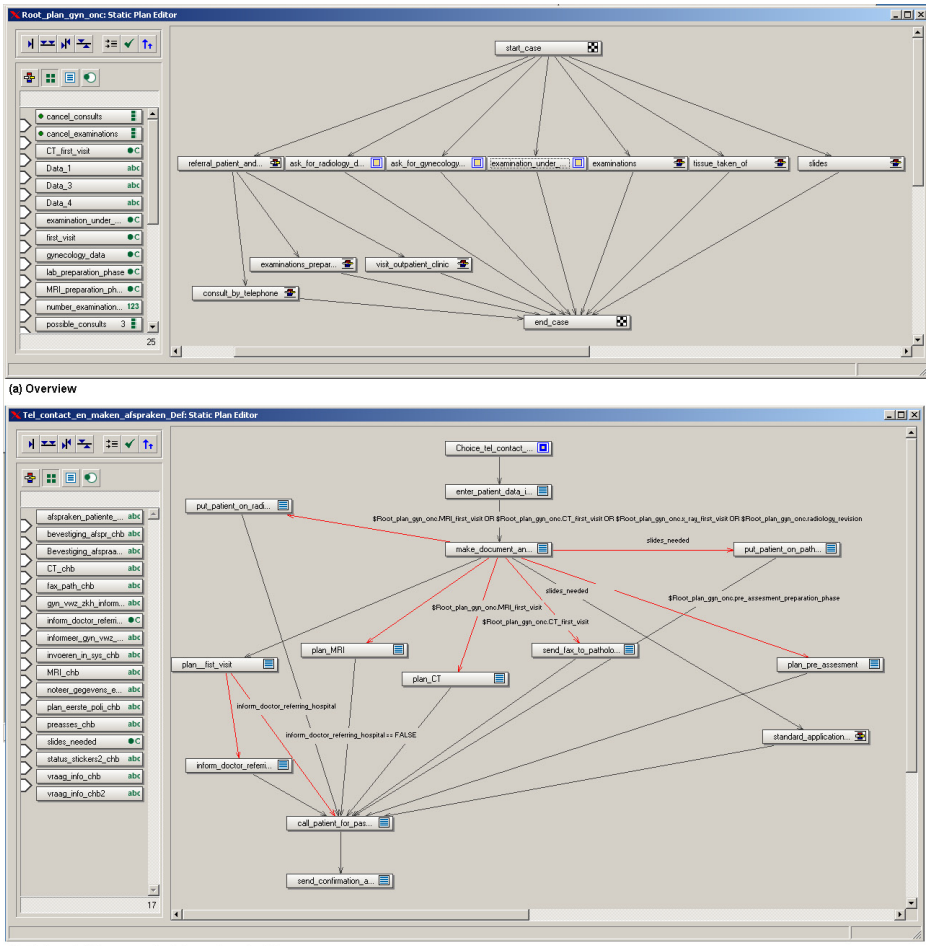

Fig. 4. Screenshots of models in the FLOWer editor.

4.3 ADEPT1

ADEPT1 is an academic prototype workflow system [35], developed at University of Ulm, Germany. ADEPT1 supports *dynamic change* which means that the process model for one individual case can be adapted. In doing so, it is possible to deviate from the pre-modeled process template (skipping of steps, going back to previous steps, inserting new steps, etc.) in a secure and safe way. That is, the system guarantees that all consistency constraints (e.g., no cycles, no missing input data when a task program will be invoked) which have been ensured prior to the dynamic (ad hoc) modification of the process instance are also ensured after the modification. The intention of the next version (ADEPT2) is to provide

full support for changes, including the propagation of process schema changes to already running instances [13].

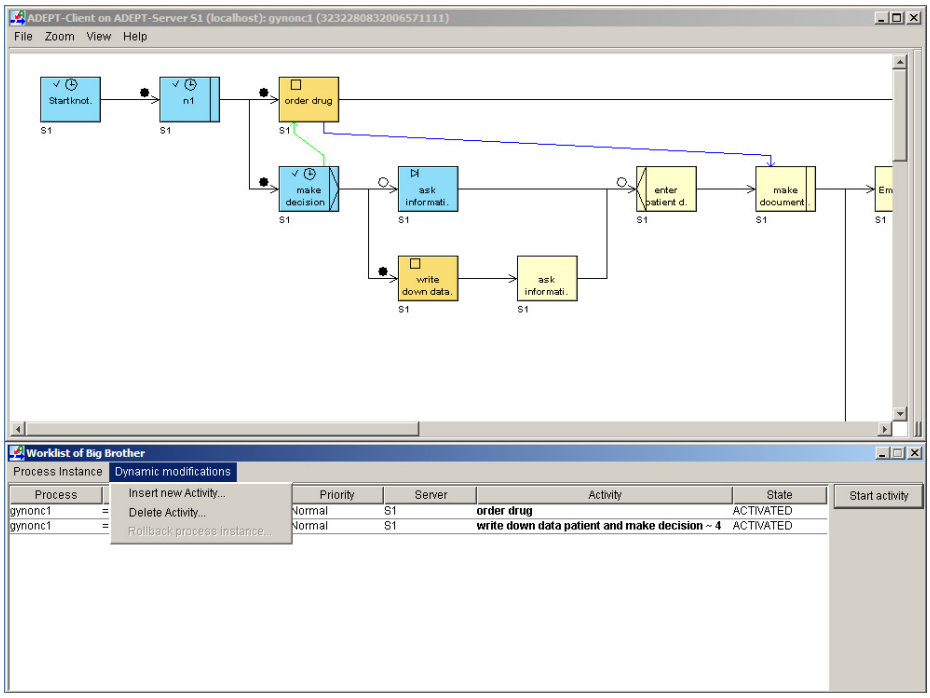

Fig. 5. Screenshot of the ADEPT1 client which shows the changed model and the worklist. AND-splits/joins are represented by a black rectangle in a node and XOR-splits/joins are represented by a black triangle in a node.

To this end, in Figure 5, we see how the first part of the gynecological oncology process is mapped to the ADEPT1 language. As ADEPT1 only supports simple data types, like string or integer, we were only able to model 10% of the process, which corresponds to the first part of the process.

In the ADEPT1 language, activities are represented by rectangles. However, as we only have XOR/AND-split/join constructs, we need to introduce dummy activities which are not executed by users. For example, the "make stickers and document" activity in ADEPT is an AND-split and the "make decision" activity is an XOR-split.

The activity "order drug" has been dynamically added to the model by selecting the option "insert new activity" from the "Dynamic modifications" menu in the worklist. More specifically, the activity "order drug" has been inserted after activity "make decision" and before activity "make document and stickers", which allows for ordering a drug in between activities "make decision" and "make document and stickers" and which is also reflected in the worklist. Note that it

is checked by ADEPT1, whether we are allowed to dynamically add the activity or not.

4.4 Declare

Declare is another academic prototype workflow system focusing on flexibility [32, 31]. In Declare the language used for specifying processes, called *ConDec*, is a *declarative* process modeling language, which means that it specifies *what* should be done. *Imperative* process modeling languages, like YAWL and FLOWer, specify *how* it should be done, which leads to over-specified processes. By using a declarative rather than an imperative / procedural approach, ConDec aims at an under-specification of the process where workers have room to "maneuver".

The ConDec language allows for modeling and enacting dynamic processes and is based on Linear Temporal Logic (LTL). In this way, it can be specified which behavior is forbidden. Moreover, within Declare it is assumed that users already know what should be done. The users can execute activities in any order and as often as they want, but they are bound by certain specified rules.

Furthermore, Declare also supports *dynamic change*, so that the process associated with individual cases can be adapted. In Declare, this means that it is possible to deviate from the pre-modeled process template by adding or removing activities or constraints. Also, model correctness is guaranteed and it is checked by Declare whether the changes are allowed or not for the cases for which the changes are intended to be applied.

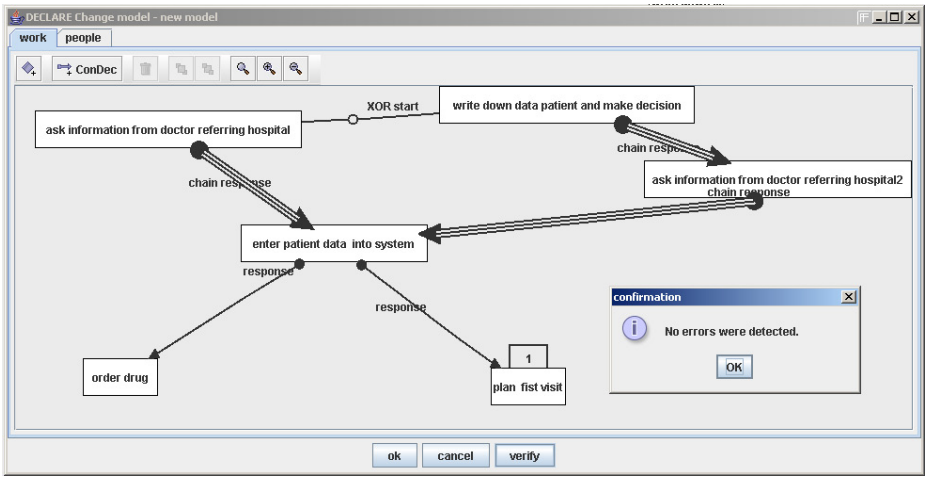

Fig. 6. Screenshot of the Declare editor.

In Figure 6, we see how the first part of the CWN of Figure 2 is mapped to the ConDec language. Also in Declare we only modeled 10% of the whole process due to the availability of only scalar data types.

In the ConDec language, the activities are represented by rectangles. Moreover, each different LTL formula, which can be used in the model, is represented by a different *template*, and can be applied to one or more activities, dependent on the arity of the LTL formula which is used. Note that the language is extensible, i.e., it is easy to add a new construct by selecting its graphical representation and specifying its semantics in terms of LTL.

For the model in Figure 6, we dynamically included the "order drug" activity and the response constraint between this activity and activity "enter patient data into system", which means that after executing "enter patient into system" there is the option to execute the "order drug" activity. After clicking on the "verify button" we get the message "No errors were detected" which means that the change is permitted.

Furthermore, in Figure 6, we see that after the "enter patient data into system" activity the activities "plan first visit" and "order drug" can be done which is indicated by a *response* arc going from the "enter patient data into system" activity to these activities. However, it is only modeled that these activities need to be done and not in which order. It is up to the user themself to decide in which order these two activities need to be done and also whether the "order drug" activity should be performed or not.

5 Evaluation

In Section 2.1 four different approaches to achieve process flexibility have been discussed. First. for the case of gynecological oncology we will discuss which flexibility approach is the best candidate for supporting the healthcare process under consideration. Following on from this, we will distinguish different kinds of healthcare processes and offer a basis for classifying their specific flexibility requirements. In this way, it becomes clear which kind of flexibility is needed for which kind of healthcare process. Finally, we use this classification to evaluate the capabilities of the offerings discussed in Section 4 in order to determine which of them can provide the best support for which kind of healthcare process.

Gynecological oncology healthcare process Clearly, the gynecological oncology healthcare process that we discussed before is an *organizational* healthcare process. Organizational processes consist of organizational tasks like medical procedures that need to be planned, appointments with different service providers that need to be scheduled, and reports that need to be written, transmitted and evaluated. Thus, a collaboration between people from different departments is a vital process characteristic. Moreover, the process is repetitive, but non-trivial. Unlike medical treatment processes, organizational processes do not provide any support for medical decision making by means of medical guidelines or medical pathways [24]. Note that also for the classification of healthcare processes we only focus on organizational processes.

The gynecological process, shown in Figure 1, is performed in an academic hospital (AMC, Amsterdam). The process deals with gynecological oncology pa-

tients for which the AMC is a reference center. As can be seen in the figure many different departments can be involved in the process of diagnosing a patient. Although in general, the art and the number of diagnostic tests is known, the total number of examinations is determined by patient characteristics and already performed diagnostic tests and the quality of these tests. To this end, it is not known beforehand or during the diagnostic process itself, which tests need to be undergone by a patient. Clearly, complex care needs to be delivered. To this end, *flexibility by underspecification* is an interesting candidate in order to provide support for the process, as it allows for the definition of an incomplete model for which the ultimate realization of tasks can be deferred until runtime. For example, the selection of tests to be performed can be made at runtime. Nevertheless, some parts of the process can be clearly defined.

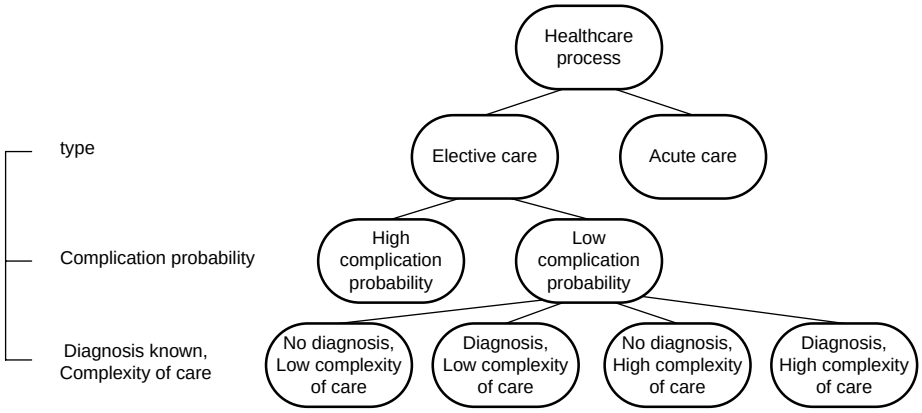

Fig. 7. Classification of healthcare processes.

Healthcare processes Next to the healthcare process discussed earlier, there exist many other (organizational) healthcare processes with totally different characteristics for which other requirements with regard to flexibility will exist. In Figure 7, we can already see which different healthcare processes will be distinguished in the sequel. As we can see in the figure only organization healthcare process will be considered, which, mainly, can be subdivided into the following two types of processes:

- *acute care* deals with critically ill patients in which patient conditions change rapidly.
- *elective care* for which it is still medically sound to postpone treatment for some days or weeks. Consequently, this kind of care can be planned in advance.

It is clear that acute care can not be planned and needs to be done in an ad-hoc fashion. Depending on the condition of the patient, which can change every minute, the right actions need to be chosen. To this end, flexibility by change is the best candidate for supporting such an ad-hoc process as the model is not fixed and can be changed into another completely specified model, thereby guaranteeing model correctness.

However, elective care can be planned in advance but still several distinct classes can be identified. The focus is on presenting a classification which covers the majority of the elective care as it is infeasible to cover everything due to the unexpected character of healthcare processes. First of all, we propose to make a distinction between healthcare processes for which the probability that *complications* arise is *high* or *low*. Depending on the patient that is dealt with, this probability can vary.

Typically, when such a complication occurs it has a high impact on the process as it requires the process to be changed dramatically in some parts. After these changes, the process needs to be made complete again so that it can be executed. The kind of flexibility needed in this situation is provided by flexibility by change. Consequently, healthcare processes for which the probability on complications is high can best be supported by flexibility by change.

Contrary, elective care for which the complication probability is low, no dramatic changes are to be expected in the process execution. Nevertheless, different classes can be identified which have their own requirements with regard to process flexibility. To this end, we propose the following dimensions: *complexity of care* and *diagnosed*. Complexity of care indicates the extent of care which is delivered to a patient, which can be either high or low. Diagnosed, indicates whether a diagnosis is known for a patient or not. In the diagnosis phase, primarily diagnostic tests are performed whereas in the treatment phase primarily treatments are performed and tests for deciding about the next steps to be done.

In case the complexity of care to be delivered is low only a few departments are involved in diagnosing and treating the patient. More or less, a standard process can be followed for diagnosis and treatment. For the diagnosis process, there is still the challenge of finally diagnosing the patient. However, as there is already an indication suggesting that low complexity of care will be required, more or less a standard process can be followed. In the case, that a diagnosis is known, the process to treat the patient will also be clear.

To this end, both the diagnostic and treatment processes can be incorporated in a complete model. However, in order to be able to capture these processes in a model, a high amount of flexibility by design is needed. Nevertheless, for most of the care processes, occasional unforeseen behavior should be anticipated, where the actual execution at runtime varies from the strict sequence implied by the process model, like the skipping of a registration step. This can be provided by flexibility by deviation.

However, the complexity of the care to be delivered can also be high. Typically, this kind of care is very specialistic care which involves special diagnostic tests and treatments in specialized centers. Moreover, collaboration between

several disciplines is needed. Diagnosing a patient can be very challenging as for some patients it can not be anticipated which diagnostic tests need to be performed. Also, when a patient is finally diagnosed a careful choice needs to be made about the next steps to be done. So, as treatments are performed it needs to be decided which additional treatments or diagnostic tests need to be undertaken and by which discipline. Such a decision also heavily depends on intermediate results of diagnostic tests, the way a patient responds to the treatment offered, and the condition of the patient themself. Clearly, for this kind of process, the ultimate realization of some parts of the model needs to be deferred until runtime. This can be provided by flexibility by underspecification.

Table 2 summarizes which flexibility approach is considered important for which kind of healthcare process. This does not imply that if a flexibility type has not been indicated for a specific type of healthcare process that is not relevant, but it is considered to be of less importance.

	flexibility by design	flexibility by deviation	flexibility by underspecification	flexibility by change
acute care				X
Elective care				
high complication probability				X
low complication probability				
low complexity, no diagnosis known	X	X		
low complexity, diagnosis known	X	X		
high complexity, no diagnosis known			X	
high complexity, diagnosis known			X	

Table 2. Flexibility needed for each kind of healthcare process.

System support Furthermore, from Table 1 it can be seen which kind of flexibility is provided by each system. If we combine these results with the results from Table 2, we can derive which system(s) can provide the best support for each kind of healthcare process.

The table shows that each flexibility type is relevant for supporting healthcare processes. For a low complex elective care process a choice needs to be made between flexibility by design and flexibility by deviation. As FLOWer supports both types, this system will be the best candidate. On the other hand, for high complex elective care processes, flexibility by underspecification is needed. To this end, YAWL will be the best candidate. Finally, for both acute care processes and processes with a high complication probability, flexibility by change

is needed. As can be seen in [28], ADEPT1 and Declare both support ad-hoc change, so they are both candidates.

As already said before, the gynecological healthcare process is an elective care process in which complex care needs to be provided. Consequently this care process can best be supported by the YAWL workflow system because of its worklet service.

6 Related Work

From the literature, it can be recognized that many workflow systems are not suitable for the healthcare domain [10, 23]. The current generation of workflow systems adequately supports administrative and production workflows but they are less adequate for healthcare processes which have more complex requirements [10]. In addition, in [33, 34], it has been indicated that so called "careflow systems", systems for supporting care processes in hospitals, have special demands with regard to workflow technology. One of these requirements is that flexibility needs to be provided by the workflow system [25, 40]. Unfortunately, current WfMS significantly fall short with regard to providing flexibility, which is also seen as a problem in the literature [5, 7, 8, 14, 20, 37]. Also, once a workflow-based application has been configured on the basis of an explicit process model, the execution of related process instances tends to be rather inflexible [1, 35, 38, 43]. Consequently, its lack of flexibility has significantly limited the application of workflow technology. The workflow systems that we chose in this paper (YAWL, FLOWer, ADEPT1 and Declare) allow for more flexibility than classical workflow systems.

Moreover, with regard to the requirements for applying workflow technology in the healthcare domain, in [17] it is indicated that real time patient monitoring, detection of adverse events, and adaptive responses to breakdown in normal processes is needed. As adaptive workflow systems are rarely implemented, this makes current workflow implementations inappropriate for healthcare [41]. Also, [12, 15, 26, 25, 40] stress that workflow systems have to support dynamic adaptation of running workflows to handle the flexibility of healthcare (therapy) processes. In case of breakdowns, managing exceptions is unavoidable [16]. Furthermore, in a real clinical setting, it is a critical challenge for any workflow management system that it is able to respond effectively when exceptions occur [30].

Furthermore, another significant gap is that no support is provided for the multidisciplinary nature of healthcare processes. In [42, 18, 39], the processes for only one department in a hospital are supported by a workflow management system. So, a requirement is that support needs to be provided for the support of cross-departmental healthcare processes which is stressed in [24, 12, 40]. Finally, a completely different requirement is that autonomous, independently developed applications need to be integrated in workflow processes and this is a risky, costly and time-consuming activity [23].

7 Conclusions

In this paper, we have investigated the flexibility requirements that need to be satisfied by workflow management systems in order to support organizational healthcare processes. First of all, a taxonomy of flexibility is presented which identifies four different approaches for achieving process flexibility for the control flow perspective. The four flexibility types, that make up the proposed taxonomy, differ with respect to the moment and the manner in which both foreseen and unforeseen behavior can be handled in a process. As a running example, we used the AMC's gynecological oncology healthcare process consisting of more than 300 different activities which has been implemented in four different workflow systems, each of which demonstrates some form of flexibility support. For this process, we identified that flexibility by design and flexibility by underspecification need to be provided, which both can best be provided by YAWL.

Furthermore, we identified different types of healthcare processes which each have their own requirements with regard to flexibility. Our results, demonstrate that all flexibility types are useful for supporting specific types of care processes. Second, individual systems tend to exhibit a degree of specialization in their approach to process flexibility, which has as consequence that different systems need to be used in conjunction with each other in order to fully support all care processes. In order to promote the use of workflow management in hospitals, the focus should be on enhancing existing tools and/or development of new ones for providing a greater support for flexibility.

A limitation of our approach is that only one healthcare process has been considered. Future research can focus on implementing healthcare processes with various characteristics in several workflow systems so that deeper insights can be gained in the process flexibility requirements.

In this paper, we only focussed on the control flow perspective of care process. One line of research would be to investigate what the flexibility requirements are when looking at other perspectives, such as the data, resource and application perspectives.

References

1. W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Workflow Modeling using Proclets. In O. Etzion and P. Scheuermann, editors, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 198–209. Springer-Verlag, Berlin, 2000.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2002.
3. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
4. 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.
5. W.M.P. van der Aalst and S. Jablonski. Dealing with Workflow Change: Identification of Issues and Solutions. *International Journal of Computer Systems, Science, and Engineering*, 15(5):267–276, 2000.

6. 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 Paper. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 22–39. Springer-Verlag, Berlin, 2005.
7. 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.
8. 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.
9. M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Dynamic. extensible and context-aware exception handling for workflows. In F. Leymann F. Curbera and M. Weske, editors, *Proceedings of the OTM Conference on Cooperative Information Systems (CoopIS 2007)*, volume 4803 of *Lecture Notes in Computer Science*, pages 95–112. Springer-Verlag, Berlin, 2007.
10. K. Anyanwu, A. Sheth, J. Cardoso, J. Miller, and K. Kochut. Healthcare Enterprise Process Development and Integration. *Journal of Research and Practice in Information Technology*, 35(2):83–98, May 2003.
11. Pallas Athena. *Case Handling with FLOWer: Beyond workflow*. Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
12. P. Dadam, M. Reichert, and K. Kuhn. Clinical Workflows - The Killer Application for Process-oriented Information Systems? In W. Abramowicz and M.E. Orlowska, editors, *BIS2000 - Proc. of the 4th International Conference on Business Information Systems*, pages 36–59, Poznan, Poland, April 2000. Springer-Verlag.
13. Peter Dadam, Manfred Reichert, Stefanie Rinderle, Martin Jurisch, Hilmar Acker, Kevin Göser, Ulrich Kreher, and Markus Lauer. Towards truly flexible and adaptive process-aware information systems. In Roland Kaschek, Christian Kop, Claudia Steinberger, and Günther Fliedl, editors, *UNISCON*, volume 5 of *Lecture Notes in Business Information Processing*, pages 72–83. Springer, 2008.
14. C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10 – 21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.
15. U. Greiner, J. Ramsch, B. Heller, M. Löffler, R. Müller, and E. Rahm. Adaptive Guideline-based Treatment Workflows with AdaptFlow. In K. Kaiser, S. Miksch, and S.W. Tu, editors, *Proceedings of the Symposium on Computerized Guidelines and Protocols (CGP 2004)*, Computer-based Support for Clinical Guidelines and Protocols, pages 113–117, Prague, 2004. IOS Press.
16. M. Han, T. Thiery, and X. Song. Managing Exceptions in the Medical Workflow Systems. In *Proceeding of the 28th international conference on Software engineering*, pages 741 – 750, Shanghai, China, 2006. ACM Press.
17. Y. Han, A. Sheth, and C. Bussler. A taxonomy of adaptive workflow management. In *Proceedings on CSCW-98 Workshop Towards Adaptive Workflow System*, 1998.
18. T. Wendler J. von Berg, J. Schmidt. Business Process Integration for Distributed Applications in Radiology. In *Third International Symposium on Distributed Objects and Applications (DOA'01)*, pages 10–19, Rome, Italy, 2001.

19. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
20. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Adaptive Workflow Systems*, Special Issue of Computer Supported Cooperative Work, 2000.
21. 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.
22. P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition.* John Wiley and Sons, New York, 1997.
23. R. Lenz, T. Elstner, H. Siegele, and K. Kuhn. A Practical Approach to Process Support in Health Information Systems. *Journal of the American Medical Informatics Association*, 9(6):571–585, December 2002.
24. R. Lenz and M. Reichert. IT Support for Healthcare Processes - Premises, Challenges, Perspectives. *Data and Knowledge Engineering*, 61:49–58, 2007.
25. L. Maruster, W.M.P. van der Aalst, A.J.M.M. Weijters, A. van den Bosch, and W. Daelemans. Automated Discovery of Workflow Models from Hospital Data. In C. Dousson, F. Höppner, and R. Quiniou, editors, *Proceedings of the ECAI Workshop on Knowledge Discovery and Spatial Data*, pages 32–36, 2002.
26. S. Miksch, R. Kosara, and A. Seyfang. Is Workflow Management Appropriate for Therapy Planning? In *Proceedings of EWGLP 2000*, pages 53–69, Amsterdam, 2000. IOS Press.
27. N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-29, BPMcenter.org, 2006.
28. N.A. Mulyar, M.H. Schonenberg, R.S. Mans, N.C. Russell and W.M.P. van der Aalst. Towards a Taxonomy of Process Flexibility (Extended Version). BPM Center Report BPM-07-11, BPMcenter.org, 2007.
29. Pallas Athena. *Flower User Manual.* Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
30. S. Panzarasa and M. Stefanelli. Workflow management systems for guideline implementation. *Neurological Sciences*, 27:245–249, June 2006.
31. M. Pesic, M.H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-based workflow models: Change made easy. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2007.
32. M. Pesic and W.M.P. van der Aalst. A declarative approach for flexible business processes management. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops (Proceedings BPM 2006 International Workshops*, volume 4103 of *Lecture Notes in Computer Science*, pages 169–180, Vienna, Austria, September4-7 2006. Springer.
33. 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.
34. 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.
35. M. Reichert and P. Dadam. ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

36. Manfred Reichert, Stefanie Rinderle, and Peter Dadam. Adept workflow management system. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors, *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, volume 2678 of *Lecture Notes in Computer Science*, pages 370–379. Springer, 2003.
37. 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.
38. S. Sadiq, O. Marjanovic, and M.E. Orlowska. Managing Change and Time in Dynamic Workflow Processes. *International Journal of Cooperative Information Systems*, 9(1-2):93–116, 2000.
39. M. Sedlmayr, T. Rose, T. Greiser, R. Röhrig, M. Meister, and A. Michel-Backofen. Automating Standard Operating Procedures in Intensive Care. Accepted for CaiSE2007.
40. M. Stefanelli. Knowledge and Process Management in Health Care Organizations. *Methods Inf Med*, 43:525–535, 2004.
41. J. Sutherland and W.-J. van den Heuvel. Towards an Intelligent Hospital Environment: Adaptive Workflow in the OR of the Future. In *Proceedings of the 39th Hawaii International Conference on System Sciences*, 2006.
42. 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.
43. M. Weske. Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In R. Sprague, editor, *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34)*. IEEE Computer Society Press, Los Alamitos, California, 2001.