Proclets in Healthcare

R.S. Mans[a,b,*], N.C. Russell[a], W.M.P. van der Aalst[a], P.J.M. Bakker[b], A.J. Moleman[b], M.W.M. Jaspers[c]

[a] *Department of Information Systems, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands*
[b] *Academic Medical Center, University of Amsterdam, Department of Quality Assurance and Process Innovation, Amsterdam, The Netherlands*
[c] *Department of Medical Informatics, Academic Medical Center, University of Amsterdam, Amsterdam, The Netherlands*

[*] Corresponding Author. Information for sending proofs: email-address:r.s.mans@tue.nl, Address: Eindhoven University of Technology, School of Industrial Engineering, Subdepartment of Information Systems (IS), Paviljoen K0.03, De Lismortel 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, fax: +31 40 243 2612

Email addresses: `r.s.mans@tue.nl` (R.S. Mans), `n.c.russell@tue.nl` (N.C. Russell), `w.m.p.v.d.aalst@tue.nl` (W.M.P. van der Aalst), `p.j.bakker@amc.uva.nl` (P.J.M. Bakker), `a.j.moleman@amc.uva.nl` (A.J. Moleman), `m.w.jaspers@amc.uva.nl` (M.W.M. Jaspers)

Abstract

Healthcare processes can be characterized as weakly-connected interacting light-weight workflows coping with different levels of granularity. Classical workflow notations fall short in supporting these kind of processes. Although these notations are able to describe the life-cycle of individual cases and allow for hierarchical decomposition, they primarily support monolithic processes. However, they are less suitable for healthcare processes. The Proclets framework is one formalism that provides a solution to this problem. Based on a large case study, describing the diagnostic process of the gynecological oncology care process at the Academic Medical Center (AMC), we identify the limitations of "monolithic workflows". Moreover, by using the same case study, we investigate whether healthcare processes can be described effectively using Proclets. In this way, we provide a comparison between the Proclet framework and existing workflow languages and identify research challenges.

Key words: Healthcare, Workflow Execution, Business Process

1. Introduction

In healthcare organizations, such as hospitals, many complex, non-trivial processes are performed which are lengthy in duration. These processes are *diverse*, *flexible* and often involve *several medical disciplines* in diagnosis and treatment. For a group of patients with the same condition, a number of different examinations and treatments may be required and the order in which they are conducted can vary greatly.

For these healthcare processes, information technology offers new possibilities for quality improvement [1]. However, in order to be able to provide this support, a distinction needs to be made between two kinds of processes. First of all, *organizational* processes capture the organizational knowledge which is necessary to coordinate interoperating healthcare professionals and organizational units (e.g. reporting of results or preparations for surgery) [1]. Based on process definitions, Workflow Management Systems (WfMSs) are able to manage the flow of work in these processes such that individual workitems are done at the right time by the proper person [2–5].

Conversely, *medical treatment* processes capture the diagnostic and therapeutic procedures to be carried out for a particular patient [1]. In this context, Computer Interpretable Guidelines (CIGs) are used which can be considered to be frameworks for specifying these kinds of processes and for standardizing them [6]. Languages for capturing CIGs facilitate clinical decision-making and have (often complex) features for standardizing medical concepts, expressing decision criteria and linking with electronic health records (see e.g. [7–11]).

Computer-interpretable Task-Network Models (TNMs) define a network of tasks that unfold over time [7, 12]. Most languages for modeling CIGs are TNMs [7, 13]. Moreover, most languages for WfMSs may also be considered as a kind of TNM [14]. However, there are important differences. A CIG language is a TNM for a *clinical* care process that realizes a *clinical/medical* goal, for example increasing doctor's adherence to an evidence-based medical guideline to optimize decisions concerning disease management. These CIG languages thus contain medical knowledge and evidence, and are *decision* oriented. In that way, their primary goal is to provide doctors with a patient-specific advice concerning diagnostic and therapeutic decisions [1, 6, 15–18]. Such computer-based guideline systems have been developed for a myriad of clinical issues, including management of heart disease [17], hypertension [18], acute myocardial infarction [19], and mechanical ventilation of patients [20]. So, such a model should be seen from the viewpoint of a physician dealing with a patient that needs to be treated for a certain illness [6]. In contrast, a workflow model is a TNM for a business process that realizes a *business* objective, for example the tuning of healthcare (logistic) processes with the aim with the aim of improving the efficiency and quality of the described process. So, such a model should be seen from the viewpoint of a manager or an analyst [6]. Moreover, a workflow model specifies *all allowable process paths* and all decision points. However, the decision points themselves need to be realized using dedicated software (e.g., expert systems based on CIGs) or medical professionals. Or, in other words, a workflow model is concerned with the logistics of work processes, not with the contents of individual tasks. Clearly, no support is provided with regard to the selection of process paths unless routing can be done on the basis of data elements. This is in contrast to guidelines where patient related data is used in order to make the right clinical decision. In order to provide optimal support for healthcare processes, we believe that effective support for both organizational and medical treatment processes is necessary and that both areas can and should complement each other.

In this paper, we *focus on the support of organizational processes by WfMSs*. WfMSs have been widely and successfully applied in various industries to streamline process execution, lower cycle times, and to monitor task completion [21–25]. Although these advantages make the application of this kind of technology in the healthcare domain interesting, its 'widespread' adoption and dissemination is the exception rather than the rule [1, 26, 27]. This is probably related to the fact that contemporary WfMSs have difficulties dealing with the dynamic nature of processes [28]. One of the main problems is that they require that the complete workflow is described as *one* monolithic overarching workflow. This assumes that a workflow process can be modeled by specifying the *life-cycle of a single case in isolation*. For real-life healthcare processes this assumption can not be made. As a result, the control-flow of several cases need to be artificially squeezed into a single model where essential parts of the control-flow are ultimately hidden inside custom-made application software or are not taken into account at all. As will be demonstrated in this paper, if a complex healthcare process is described in this way, this results in an unreadable process definition.

This can be illustrated when considering a typical healthcare process for the diagnosis of patients. In general for a patient this consists of multiple visits to a hospital in order to meet with doctors and undergo diagnostic tests (e.g. a lab test). However, there also steps in which several medical specialists meet in order to discuss the status of patients. Clearly, some tasks may operate at the level of a single patient, whereas other tasks operate at the level of a group of patients. So, processes may rely on information that is at different levels of *aggregation.* Note that this is very different from the classical notion of hierarchy in workflows as aggregation cuts across multiple cases.

The process of diagnosing a patient typically consists of the execution of a number of smaller processes that run in conjunction with each other. Flexibility in healthcare processes is needed because these small processes can be instantiated and synchronized at any point in time. For example, at any point in the process of diagnosing a patient, a doctor may order a lab test. However, although these process fragments execute independently from each other, a certain "magnetic force" exists between them. Such process fragments can best be characterized as *weakly-connected interacting lightweight workflows.*

To date, contemporary WfMSs do not offer support for weakly-connected interacting lightweight workflows which can deal with information that is at varying levels of aggregation. An interesting means of solving this issue is provided by *Proclets* [28, 29]. Proclets are a framework modeling and enacting lightweight workflow processes. Together with *performatives* and *channels* it is possible to describe how these Proclets *interact* with each other. Moreover, the interaction between these Proclets is modeled explicitly using structured messages, called performatives, which are exchanged via channels.

Proclets provide an interesting means of modeling and executing a healthcare process in WfMSs, and can assist in realizing the promised benefits of applying WfMSs. In this paper *we investigate whether organizational healthcare processes can indeed be modeled using this technique and how this compares to existing workflow approaches.* In order to do so, we take the following approach. We focus on the gynecological oncology workflow as it is performed at the Academic Medical Center (AMC) in Amsterdam, a large academic hospital in the Netherlands, which is considered to be *representative* of other healthcare processes. The selected healthcare process describes the diagnostic process for patients visiting the gynecological oncology outpatient clinic and is a large process consisting of around 325 activities. In earlier work, this healthcare process has already been modeled in full using the workflow languages, YAWL and FLOWer, and has been partially modeled using the Declare and ADEPT1 workflow languages [30]. *This allows us to investigate the problems existing workflow approaches face when modeling this type of process.* We discuss in detail how the healthcare process has been modeled using the YAWL workflow language. For FLOWer, ADEPT1, and Declare, we summarize the issues encountered. This leads on to a discussion of how the same healthcare process can be modeled using Proclets and how the identified issues can be addressed. It is worth noting that the reason for implementing a hospital process in the four workflow systems mentioned above was to identify the requirements that need to be fulfilled by workflow systems in order to be successfully applied in a hospital environment. These requirements have been discussed in [31].

This paper is structured as follows: Section 2 introduces the Proclets approach. In Section 3, we introduce the gynecological oncology healthcare process and discuss how it is modeled using YAWL, FLOWer, Declare, and ADEPT1. In Section 4, we discuss the modeling of the healthcare process using Proclets and elaborate on how the limitations, mentioned in Section 3, are addressed. Related work is outlined in Section 5. Section 6 discusses the experiences associated with modeling healthcare processes using Proclets. Finally, the paper is concluded in Section 7.

2. Introduction to Proclets

In this section, we discuss how Proclets provide a framework for modeling workflows. The concepts of the framework have already been introduced in [28, 29]. In this section, we give an introduction to this framework in order to assist the reader in better understanding the Proclet models that will be shown in the remainder of this document. For complete details we refer the reader to [28, 29]. At the end of this section, the use and operation of Proclets will be illustrated by a small healthcare example.

In Figure 1, a graphical representation of the concepts, which underpin the framework, is shown. As can be seen, there are five main concepts each of which will be discussed below.

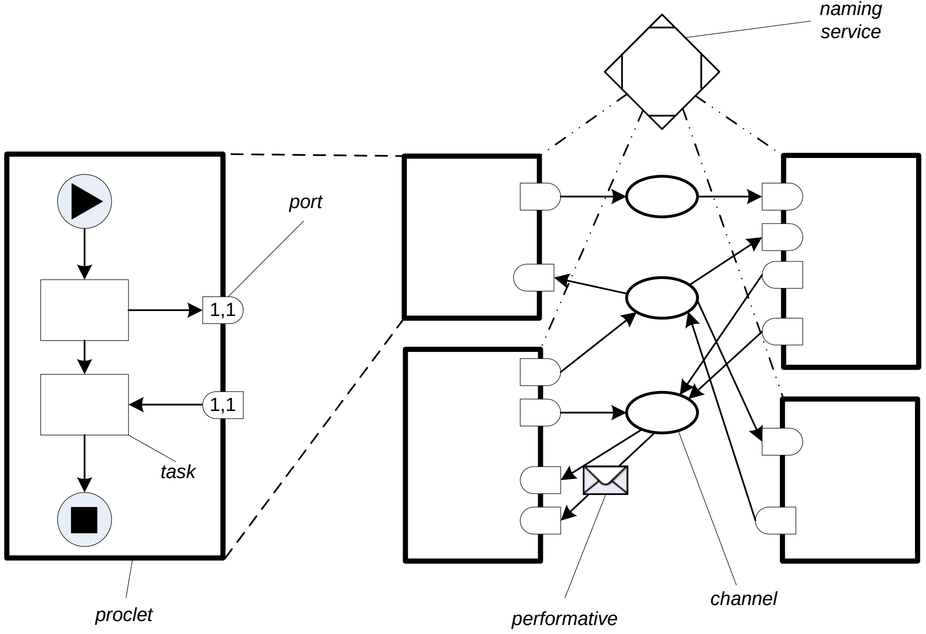


Figure 1: Graphical representation of the Proclet framework.

The framework is centered around a ***Proclet***. There is a distinction between a Proclet class and a Proclet instance. A *Proclet class* can best be seen as a process definition which describes which tasks need to be executed and in which order. For a Proclet class, instances can be created and destroyed. One instance is called a *Proclet instance*. For the definition of a Proclet class, a selection can be made between multiple graphical languages. In this paper, we use a graphical language based on the YAWL language [32]. However, other languages, such as Petri Nets [33] or EPCs [34], can also be used. With regard to the selection of a graphical language, some limitations apply. First of all, Proclet instances need to have a state and they need to support the notion of a task. Second, a Proclet class needs to be sound [35].

With regard to the communication and collaboration among Proclets, so called *channels*, *ports*, and *performatives* are important. First of all, Proclets interact with each other via ***channels***. A channel can be used to send a ***performative*** to an individual Proclet or to a group of Proclets. A performative is a specific kind of message with several attributes which is exchanged between one or more Proclets. A performative has the following attributes:

- *Time*: the moment the performative was created/received.
- *Channel*: the medium used to exchange the performative.
- *Sender*: the identifier of the Proclet creating the performative.
- *Set of receivers:* the identifiers of the Proclets receiving the performative, i.e. a list of recipients.
- *Action*: the type of the performative.
- *Content*: the actual information that is being exchanged.

The role of the action attribute deserves some special attention. This attribute can be used to specify the illocutionary point of the performative. The five illocutionary points identified by Searle [36] (assertive, directive, commissive, declarative, expressive) can be used to specify the intent of the performative. Examples of typed performatives identified by Winograd and Flores are request, offer, acknowledge, promise, decline, counter-offer, and commit-to-commit [37] which each represents a change in the state of a conversation. In the model no restriction is made to any single classification of performatives (i.e. a fixed set of types). It is important to use the experience and results reported by researchers working on the language/action perspective [37] as these give an insight into the broader requirements in this area.

Of course, it is possible to add more attributes to a performative. Note that a channel may have different properties which affect the sending and receiving of performatives, e.g. push/pull or synchronous/asynchronous. In order for Proclets to be able to find each other there is a ***naming service*** which keeps track of existing Proclets. A Proclet class and instances of it are defined in the following way:

- A Proclet class has a **unique name**. In the same way, an instance of a Proclet class has an unique identifier.

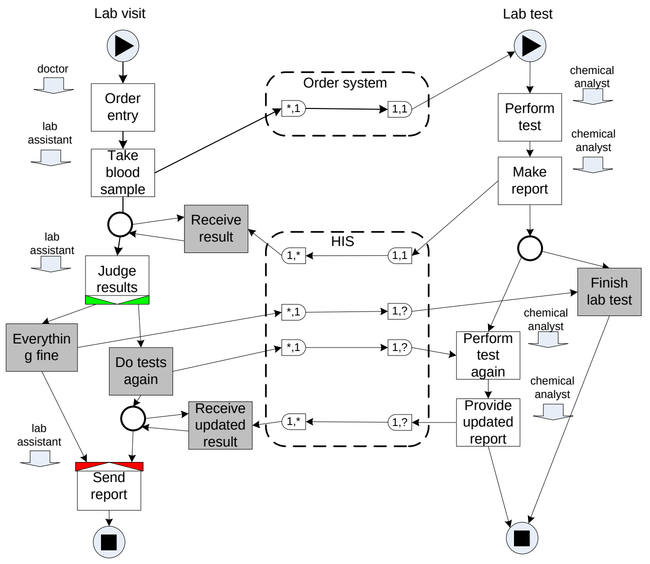
- A Proclet class has ***ports***. Performatives are sent and received via these ports in order for a Proclet to be able to interact with other Proclets. Every port, either incoming or outgoing, is connected to one task. Moreover, a port has two attributes.

  First, the *cardinality* specifies the number of recipients of performatives exchanged via the port. An $*$ denotes an arbitrary number of recipients, $+$ at least one recipient, 1 precisely one recipient, and ? denotes no or just one recipient. Note that by definition an input port has cardinality 1.

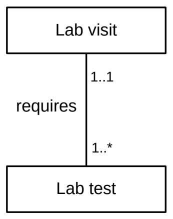
  Second, the *multiplicity* specifies the number of performatives exchanged via the port during the lifetime of an instance of the class. In a similar fashion to the cardinality, an $*$ denotes that an arbitrary number of performatives are exchanged, $+$ at least one, 1 precisely one, and ? denotes that either one or no performatives are exchanged. Note that by definition an input port has a multiplicity of 1 or ?.

- A Proclet instance has its own **knowledge base**. Knowledge in the knowledge base can be used to make *routing decisions.* This knowledge can range from simple data to beliefs about other Proclets. Building a good knowledge base is not a trivial task. First of all, there has to be an ontology to characterize the intended meaning of terms and concepts. Then, the scope and knowledge acquisition process have to be identified. When defining the knowledge base, well known notions from the medical domain can be used. For example, notions from the Open Knowledge Base Connectivity (OKBC) knowledge model can be used [38].

  Here, we use a more restrictive definition of a knowledge base (unlike the original definition in [28, 29]). Each Proclet instance has its own knowledge base for storing performatives that are received and sent. Parts of the knowledge base can be public or private. The public part is identical for all instances of the class, i.e. this part resides at the class level even though it holds information about instances. The private part resides exclusively at the instance level.



(a) Two proclet classes connected through two channels



(b) Class diagram containing the two proclet classes

Time	Channel	Sender	Receivers	Action	Content	Scope	Direction
11:00	Order system	Lab visit - John	Lab test – HGB John	Create	Can you perform a HGB test for John?	Private	OUT

(c) Example of a performative

Figure 2: Example of two Proclet classes.

- The knowledge base can be queried by tasks. A task may have a *precondition* based on the information that can be found in the knowledge base. A task can only fire if (1) the task in the net itself is enabled, (2) each input port contains a performative, and (3) the precondition evaluates to true. Note that for the YAWL language, as can be seen in Figure 2, multiple ports can be connected to an input condition. In this case, an instance is created on the receival of each performative.

- A task connected to an output port may have a *postcondition*. The postcondition specifies for the output ports, the number of performatives generated and their content. The postcondition may also depend upon information that can be found in the knowledge base.

In order to illustrate the framework, we use the small healthcare-related example shown in Figure 2(a). The example shows an organizational process which deals with the process in which a doctor orders a selection of lab tests for a patient. Afterwards, the patient visits the lab where a blood sample is taken. For the sample, several lab tests are performed and the final outcomes of the tests are reconciled in a report. There are two Proclet classes. The Proclet class “lab visit” is instantiated for every patient who visits the lab for whom a blood sample is taken. Proclet class “lab test” is instantiated for every lab test that needs to be performed on the blood sample. Hence, there is an one-to-many relationship between “lab visit” and “lab test” as shown by the relationship *requires* in the class diagram in Figure 2(b). Note that the white colored tasks are executed by a human resource and the grey colored tasks are executed automatically. For the white colored tasks it is indicated which role is required when performing the task.

For the “lab visit” Proclet, first a doctor orders a lab examination for a patient and indicates which tests are required (“Order entry” task). When a patient visits the lab, a blood sample is taken by a lab assistant (“Take blood sample” task). Upon completion of the task, a trigger for each required lab test is initiated, so that for every lab test a single instance of the Proclet class “Lab test” is created. Consequently, the cardinality of the outgoing port of the “Take blood sample” is $*$. Moreover, the multiplicity is 1 which means that during the lifetime of an instance of the class “Lab visit” exactly one performative is sent via this port. The creation performative is sent via the lab order system, which explains why the name of the channel is “Order system”. The input port connected to the input port of the “Lab test” Proclet class has cardinality 1 and multiplicity 1 as an instance can only be created once. Figure 2(b) shows an example of a performative that is sent by a “Lab visit” Proclet to a “Lab test” Proclet. From the figure, we can see that at 11 ’o clock a performative is sent by the “Lab visit” Proclet for patient John in order to create an instance of the “Lab test” Proclet called “Lab test - HGB John”. More specifically, an instance of a “Lab test” Proclet class is created so that a hemoglobin (HGB) blood test can be performed. The performative is stored in the private knowledge base of the “Lab visit” Proclet.

After an instance of the “Lab test” Proclet class has been created, a chemical analyst performs a test on the blood sample (“Perform test” task) followed by the creation of a report which contains the result of the test (“Make report”). Note that the performance for a specific lab test could be, for example, based on a link to a code from a controlled medical terminology. As a consequence of performing the “Make report” task, a performative is sent to the instance of the initiating Proclet class “Lab visit”. Note that each instance of “Lab test” sends performatives via the Hospital Information System (HIS). The results of the individual lab tests are received by the “Receive result” task. The input port of task “Receive result” has cardinality 1 and multiplicity $*$, indicating that reports of multiple tests may be received. Each performative received is stored in a knowledge base. The “lab visit” Proclet inspects this knowledge base continuously to determine whether a report for each initiated lab test has been received so that the “Judge results” task can be performed. This means that for this task a precondition has been defined which evaluates to true once a report has been received for each lab test that was initiated.

Based on the reports received, a lab assistant decides whether any of the lab tests need to be performed again or that everything is fine (i.e. were valid results obtained or did any exceptions occur). If the tests do not need to be done again, the “Everything fine” task is performed after which all instances of the “Lab test” Proclet class are destroyed via the “Finish lab test” task. In the situation where the tests need to be done again, a performative is sent via the “Do tests again” task to the “Perform test again” task to all “Lab test” Proclet instances to indicate that the lab test needs to be redone. Note that the cardinality of the output port of both the “Do tests again” and “Everything fine” tasks is $*$, i.e., in a single step all “lab test” Proclets are informed whether the tests need to be redone or not. Moreover, the ports connected to the “Perform test again” task and “Finish lab test” task both have cardinality 1 (i.e. one recipient) and multiplicity ? (one performative is sent via one of the two ports). After that a chemical analyst performs the test again, the “Provide updated report” task is performed which sends the updated report to the “Lab visit” Proclet instance where all of the results are collected via the “Receive updated result” task. Finally, the doctor, who requested the lab examination, is informed via the “Send report” task after which the “Lab visit” Proclet instance is destroyed.

The example in Figure 2 is rather simplistic and hides many details. For example, the two Proclet classes are not general enough in the sense that they do not handle the problems that may relate to the results of tests. However, it compactly illustrates the main features of Proclets. For full details on the formalism we refer the reader to [28, 29]. Note that as the example shows an organizational process, it does not contain any logic / data for making a clinical decision. So, for the “Order entry” and “Judge results” tasks there is respectively no guidance in the selection of lab tests and whether to do the tests again or not. However, we foresee that CIGs could provide guidance in such a decision process.

3. Limitations of Monolithic Workflows

In this section, we identify the problems that existing monolithic workflow approaches are facing when dealing with the dynamic nature of processes. In order to do this, we take the following approach. First, we examine the gynecological oncology workflow as it is performed at the Academic Medical Center (AMC) in Amsterdam which is considered to be representative of other healthcare processes. In previous work this process has been modeled in full using two workflow languages, YAWL and FLOWer [30], and has been partially modeled using the Declare and ADEPT1 workflow languages [30]. We discuss the selected healthcare process in detail by elaborating on how it has been modeled using the YAWL workflow language and identify the issues that arose when doing so. For FLOWer, ADEPT1, and Declare, we also discuss the issues that arose when implementing the process although we do not elaborate on specific implementation details. In doing so, we exemplify the problems existing workflow approaches are facing. Subsequently, in Section 4, we discuss how the same healthcare process is modeled using Proclets and how the issues identified can be addressed using our approach.

The gynecological oncology workflow is a large process, consisting of over 325 activities, and is performed at the gynecological oncology outpatient department at the AMC hospital. The AMC is the most prominent medical research center in the Netherlands and one of the largest hospitals in the country. The healthcare process deals with the diagnosis of patients suffering from cancer once they are referred to the AMC hospital for treatment. The care process can be considered to be non-trivial and illustrative of other healthcare processes, both at the AMC and in other hospitals.

The healthcare process under consideration consists of two distinct parts. The first is depicted in Figure 3 and shows the top page of the YAWL model. The process describes all of the steps that may be taken with a patient up to the point where they are diagnosed. The process starts with the "referral patient and preparations for first visit" composite activity. This subprocess deals with the steps that need to be taken for the first visit of the patient to the outpatient clinic. The next step in the process is the "visit outpatient clinic" composite activity where the patient visits the outpatient clinic for a consultation with a doctor. Such a consultation can also be done by telephone ("consultation by telephone" composite activity). During a visit or consultation, the patient discusses their medical status with the doctor and it is decided whether any further steps need to be taken, e.g., diagnostic tests.

The execution of the tests that may be needed are modeled by the "examinations" multiple instance task which allows for the concurrent instantiation of a number of different tests for a patient. However, for each patient there are also other steps that may be taken. These are modeled by the "ask for gynecology data", "ask for radiology data", and "examination under anesthetic" composite tasks and the "ask for pathology slides" and "take tissue sample" tasks. For example, the "ask for pathology slides" and "take tissue sample" tasks model the situation where a pathology examination is required after which the referring hospital is requested to send their pathology slides to the AMC or a tissue

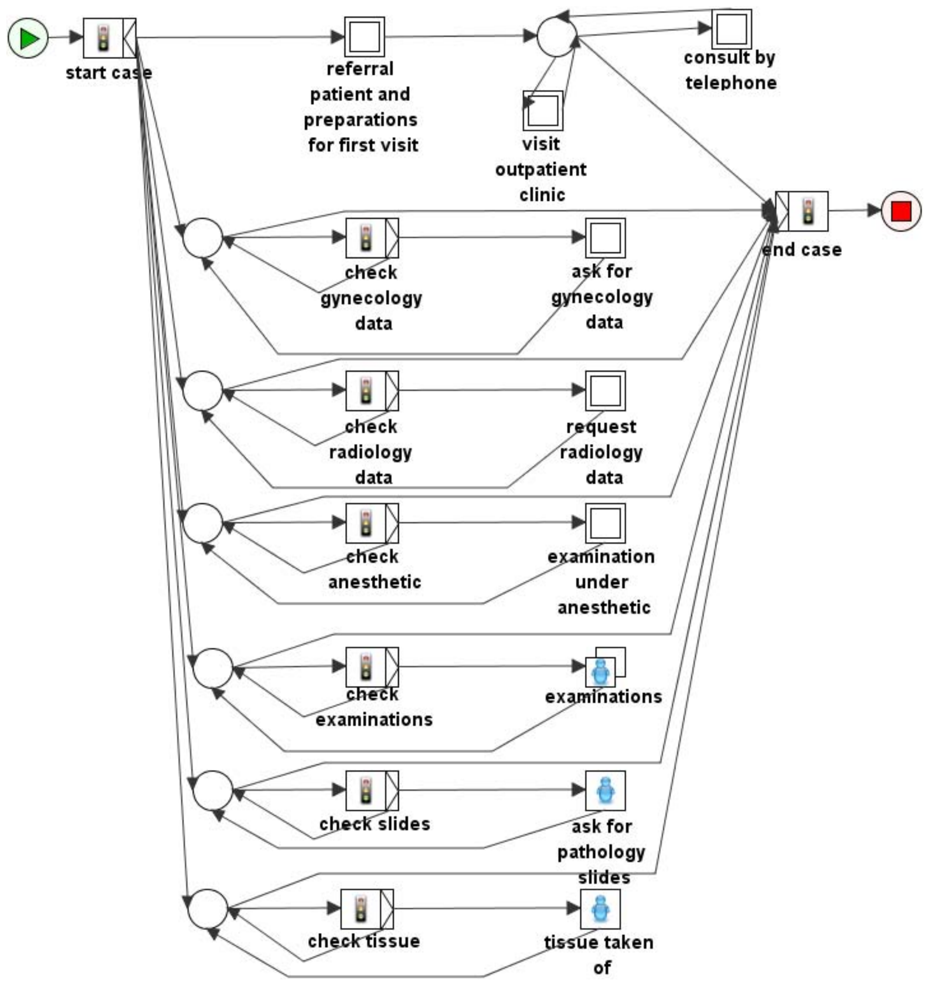


Figure 3: General overview of the gynecological oncology healthcare process.

sample is taken at the AMC.

Looking at the overall process we see that while the patient is visiting the outpatient clinic (shown in the top part of Figure 3) it is possible for a series of subprocesses to run concurrently (as shown in the lower part of the figure). As the execution of these subprocesses can be complex and time consuming, there is no guarantee that all of them will be finished before the start of the next patient consultation, e.g. the result of a certain test might be delayed. Consequently, these subprocesses should be seen as separate inter-twined life-cycles running

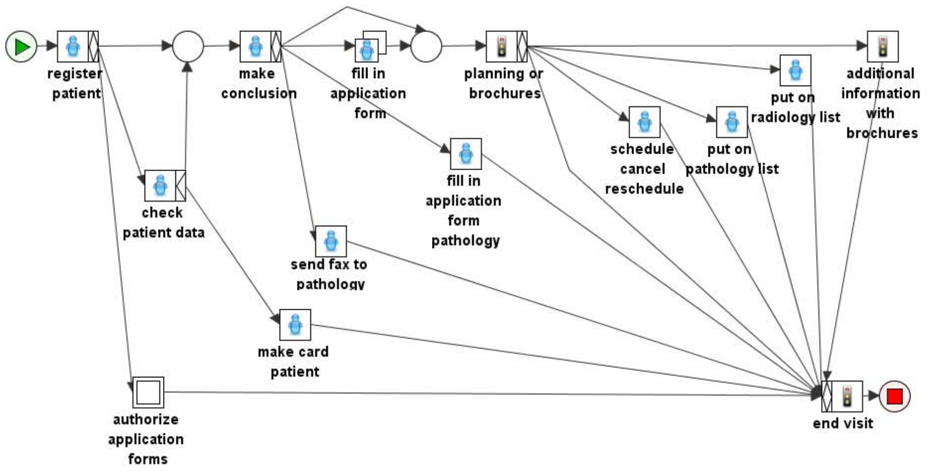


Figure 4: Visit of the patient to the outpatient clinic.

at different speeds rather than as one workflow covering different but related cases. However, if we want to denote that there is in fact a connection between these related cases, we need to model them in one monolithic workflow. For the FLOWer, Declare, and ADEPT1 workflow languages, these observations also apply. Therefore, we can conclude that for existing workflow approaches *cases need to be straightjacketed into a monolithic workflow despite the fact that it is more natural to view processes as inter-twined loosely-coupled object life-cycles.*

In Figure 4, the subprocess underlying the "Visit outpatient clinic" composite task is shown which describes the visit of a patient to the outpatient clinic. During such a consultation, the medical status of the patient is discussed and a decision is made about the next steps to be taken ("Make diagnosis" task). At different stages during the process, several administrative tasks, such as handing out brochures (task "Additional information with brochures"), and producing a patient card (task "Make patient card") may be necessary. As a result of the execution of the "Make diagnosis" task, subsequent steps in the process need to be triggered, such as further diagnostic tests or a pathology examination. These subsequent steps are depicted in the top page of the YAWL model (see Figure 3). As a consequence, they can only be enabled when the process modeled in Figure

4 is already finished. It would be more natural if these kind of processes were instantiated at the moment that it is known that they need to be created, i.e. immediately after execution of the “Make diagnosis” task. In general, for each of the subprocesses modeled in Figure 3, no *direct interaction* can take place during their execution. This is due to the fact that in YAWL there is no way of modeling interactions between (sub)processes. The same observation holds for FLOWer, ADEPT1, and Declare as well. Consequently, facilitating interactions between (sub)processes is far from trivial. Where these need to be supported, they are typically hidden in application logic or in custom built applications or even not taken into account at all. Another option would be to model the whole process in one diagram (so, no hierarchical decomposition) with the necessary interactions being modeled via case data. However, this would result in a large unreadable process model which does still not show the interactions between (sub)processes.

Note that business process notations exist which support interactions between processes. For example, the Business Process Modeling Notation (BPMN) allows the flow of messages between two entities to be shown via the message flow construct [39]. In general, we can conclude that *as most workflow languages do not provide support for interaction between (sub)processes, it is difficult to model interactions between processes.*

In Figure 5, the second part of the gynecological oncology healthcare process is shown. This involves meetings between gynecological oncology doctors and other medical specialists. First, the participants from the different medical disciplines prepare themselves for these meetings (“prepare radiology, pathology, and MDO meeting” composite task). During the radiology meeting (composite task “radiology meeting”), the doctors from gynecological oncology discuss with a radiologist the results of the radiology tests that have been performed for various patients during last week. The same holds for the “pathology meeting” composite task for the pathology examinations that have been performed during the last week. Finally, during the MDO meeting (“MDO meeting”) the medical status of patients is discussed and a decision is made about their final diagnosis

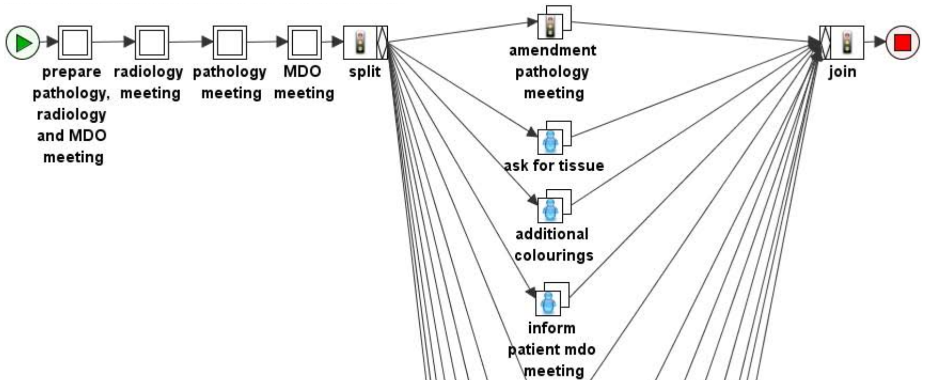

prepare pathology, radiology and MDO meeting
radiology meeting
pathology meeting
MDO meeting
split
amendment pathology meeting
ask for tissue
additional colourings
inform patient mdo meeting
join

Figure 5: Meetings which are held on Monday afternoons to discuss the medical status of patients.

before the treatment phase is started. Finally, as a result of these meetings, several subsequent steps may need to be initiated for individual patients. These steps are modeled at the right-hand side of Figure 5. For example, for some patients, existing tissue may need to be re-examined whereas for others, the referring hospital may need to be asked to send their pathology material to the AMC for investigation ("ask for tissue" task).

However, most importantly, compared to the two models discussed earlier, we are dealing with a *group of patients* instead of a single patient. Obviously, compared to the two previous models, we are dealing with a different level of *granularity* and one instance aggregates information about several other instances. Due to this difference, the workflows executed for a single patient, shown in Figure 3, and the workflow executed for a group of patients, shown in Figure 5, are modeled separately. Consequently, the two models are completely *disconnected* whereas in reality (examinations for) patients need to be registered for these meetings, which can be initiated from different places in the process described in Figure 3. For example, a patient can be registered during the initial phases of the process and also during a visit to the outpatient clinic. Should these workflows, operating at different levels of aggregation, need to be described in a single model, a decision needs to be made about what is considered to be the unit of modeling. So, is the “case” a service executed for a single patient, the illness a patient is suffering from, or a group of patients suffering from a certain illness. All of these are at different levels of granularity. The selection of a particular type of case at one of the aggregation levels causes problems because the process cannot be “straightjacketed” into a single case concept.

For the modeling of the healthcare process using the FLOWer, ADEPT1, and Declare workflow languages, the same problem applies. We are not aware of any workflow language which is able to deal with different levels of granularity. Consequently, *models often need to be artificially flattened as they are unable to account for the mix of different granularities that co-exist.* Note that the different units of modeling which are possible when modeling a workflow do not necessarily need to be aggregations of each other. Proclets can be in a “parent-child” (“one-to-many”) relationship, but also in more complex relationships like “brother-sister”, “many-to-many”. Therefore, we use the term granularity instead of aggregation.

Furthermore, the fact that multiple patients can be registered for the afore mentioned meetings (even from different points in the process) indicates that *one-to-many* relationships may exist between entities in a workflow. For example, during a visit to the outpatient clinic, a patient can be registered for discussion during an MDO meeting. This means that a one-to-many relationship exists between the entity “MDO meeting” and the “visit outpatient clinic” entity. However, as models are unable to account for different granularities that co-exist in a workflow this also means that it is impossible to capture one-to-many and many-to-many relationships that may exist between entities in a workflow. Although, *it is impossible to capture the fact that one-to-many and many-to-many relationships exist between entities in a workflow, such relationships are common as can be seen in any data/object model.*

We have discussed problems that we are faced with when modeling the gynecological oncology healthcare process using the YAWL, FLOWer, ADEPT1, and Declare workflow languages. In summary, we may conclude that existing workflow approaches currently exhibit the following problems:

- **Issue 1:** Models need to be *artificially flattened* and are unable to account for the mix of granularities that co-exist in real-life processes.
- **Issue 2:** Cases need to be *straightjacketed into a monolithic workflow* even though it is more natural to see processes as inter-twined loosely-coupled object life-cycles.
- **Issue 3:** It is impossible to capture the fact that *one-to-many* and *many-to-many* relationships exist between entities in a workflow, yet such relationships are common as can be seen in any data/object model.
- **Issue 4:** It is difficult to model interactions between processes, i.e., *interaction is not a first-class citizen* in most process notations.

Proclets, as introduced earlier, have been developed to specifically address these problems.

4. Realization of the Gynecological Oncology Workflow using Proclets

In this section, we elaborate on how the gynecological oncology healthcare process is modeled using Proclets. First, in Section 4.1, we discuss which entities can be identified in the workflow and how they relate to each other. In Section 4.2, a selection of Proclet classes will be discussed, illustrating how the entire healthcare process is modeled using Proclets. However, most importantly, it will be explained how the issues identified in Section 3 are addressed using Proclets.

4.1. Overview

The class diagram in Figure 6 gives an overview of the entities that exist within the healthcare process and the relationships between them. The dark-grey colored classes correspond to concrete Proclet classes. The inheritance relations show which Proclet classes have common features, i.e., the light-grey

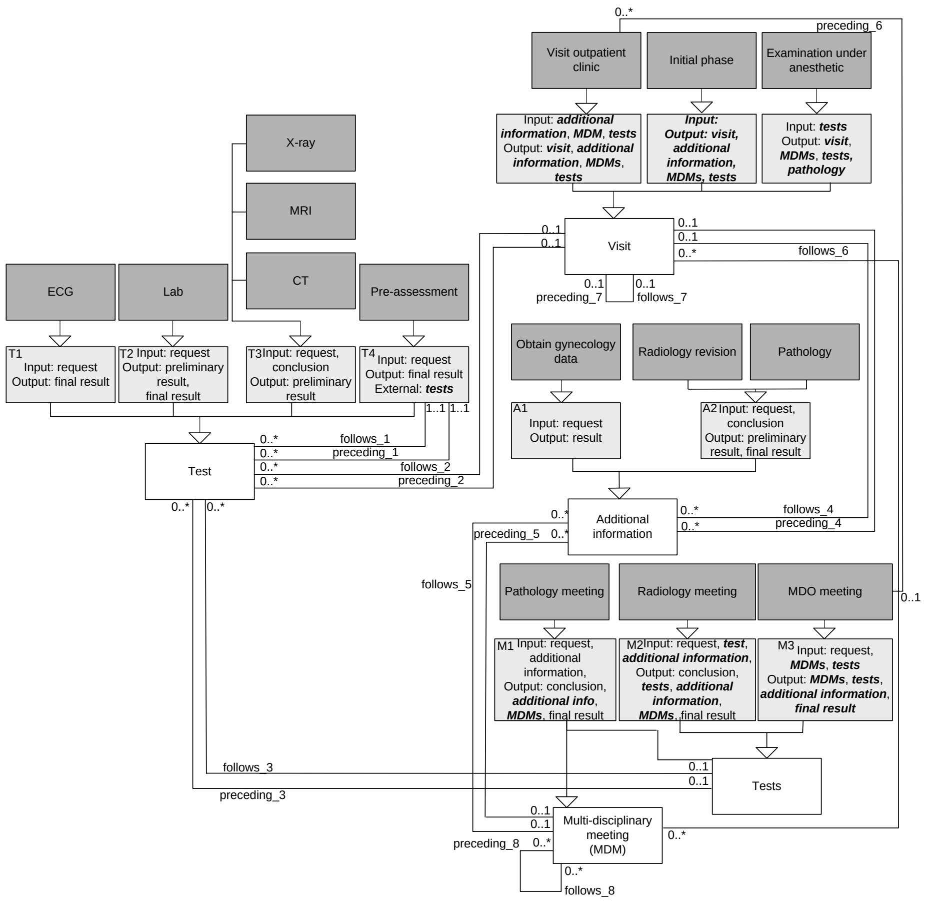

Visit outpatient clinic
Initial phase
Examination under anesthetic
Input: additional information, MDM, tests Output: visit, additional information, MDMs, tests
Input: Output: visit, additional information, MDMs, tests
Input: tests Output: visit, MDMs, tests, pathology
X-ray
MRI
CT
ECG
Lab
Pre-assessment
Visit
T1 Input: request Output: final result
T2 Input: request Output: preliminary result, final result
T3 Input: request, conclusion Output: preliminary result
T4 Input: request Output: final result External: tests
Test
Obtain gynecology data
Radiology revision
Pathology
A1 Input: request Output: result
A2 Input: request, conclusion Output: preliminary result, final result
Additional information
Pathology meeting
Radiology meeting
MDO meeting
M1 Input: request, additional information, Output: conclusion, additional info, MDMs, final result
M2 Input: request, test, additional information, Output: conclusion, tests, additional information, MDMs, final result
M3 Input: request, MDMs, tests Output: MDMs, tests, additional information, final result
Tests
Multi-disciplinary meeting (MDM)
follows_1
preceding_1
follows_2
preceding_2
follows_3
preceding_3
follows_4
preceding_4
follows_5
preceding_5
follows_6
preceding_6
follows_7
preceding_7
follows_8
preceding_8

Figure 6: Class diagram outlining the concepts that exist within the healthcare process and their relationships.

and white colored classes can be seen as abstract classes used to group and structure Proclets. The associations show the relationships that exist between Proclet classes together with their multiplicity.

Starting with the white colored classes, we see that four main entities exist within the healthcare process. These different entities have been identified by looking from a high level viewpoint at the gynecological oncology healthcare process. By taking this viewpoint, four different kinds of (sub)processes can be identified which together constitute the entire gynecological oncology healthcare process. More specifically, for the (sub)processes that belong to a main entity, process related similarities can be identified. These similarities relate to the general purpose for which these (sub)processes are executed (e.g. a test for a single patient or a meeting to discuss a group of patients). Moreover, these similarities also related to the medical departments that are responsible for these (sub)processes and whether the presence of the patient is necessary. Below, these four main entities will be discussed together with an explanation of the rationale for defining them.

- **Visit:** A patient can visit a hospital multiple times to see a doctor. The gynecological oncology department is responsible for the complete process of such a visit. A visit can either be at the outpatient clinic where the doctor examines the patient ("Visit outpatient clinic" class), or an examination under anesthetic ("Examination under anesthetic" class). Moreover, also related to a visit are the initial stages of the process ("Initial phase") in order to prepare for the first visit of the patient to the outpatient clinic.
- **Test:** A gynecological oncology doctor can select multiple diagnostic tests that need to be conducted for a patient. These tests are performed at various medical departments (other than the gynecological oncology department). The tests that can be chosen range from medical imaging ("MRI", "CT", and "X-ray" classes) to a lab test ("Lab" class), an ECG ("ECG" class), and a pre-assessment ("Pre-assessment" class). For all of these, the presence of the patient is required. Note that in principle many more diagnostic tests can be selected by a doctor. In the class diagram we only model the tests that are selected most frequently.
- **Additional information:** A gynecological oncology doctor might require additional information in order to come to a final diagnosis. This may involve requesting the hospital, that referred their patient to the AMC for treatment, to send their data to the AMC hospital so that it can be reviewed. They can be requested to send patient files (class "Obtain

gynecology data"), to send pathology slices ("Pathology" class), or to send radiology data ("Radiology revision" class). However, the "Pathology" Proclet class also involves (re)examining patient tissue which has been collected at the AMC. For all of these processes, the patient does not need to be present.

- **Multi-disciplinary meeting:** Every Monday afternoon multiple meetings are organized for discussing the status of patients and/or the outcome of examinations. For each of these meetings, the gynecological oncology department is involved. In addition, these meetings involve the departments of radiology ("Radiology meeting" class), pathology ("Pathology meeting" class) and a multidisciplinary meeting ("MDO meeting" class) involving the departments of radiotherapy, and internal medicine (in order to give chemotherapy). For these meetings, the patient does not need to be present.

The entity types mentioned above are very general. In principle, they generalize very well to other healthcare processes in which these entities may apply.

For the four main entity types, Proclets are instantiated a variable number of times and interact in different ways with each other. For these interactions between Proclets, a single Proclet might require multiple inputs and outputs from other existing Proclets. For example, a lab test can be triggered during a visit to the outpatient clinic and also during the initial phases of the process or during an MDO meeting.

To make these interaction related commonalities explicit, the light-grey colored classes in Figure 6, outline these interaction characteristics in terms of inputs and outputs. The items depicted in bold italics indicate that an interaction is optional whereas an item written in normal text indicates that an interaction is mandatory. In this way, the light-grey colored classes explicitly identify (at a high level) the interface that exists for a specific (group of) Proclet(s). Note that not all Proclet classes have the same level of aggregation. The multi-disciplinary meeting related Proclet classes all deal with a group of patients whereas the other Proclet classes are related to a single patient.

We will now elaborate on the four main types of Proclet classes that have been identified and examine their interaction with other Proclet classes. First, we focus on the "Test" and "Additional information" entities in isolation. Then, we focus on the "Visit" and the "Multi-disciplinary meeting (MDM)" entities and elaborate on the associations with other entities. In general, an association with the name "follows" indicates that, seen from the viewpoint of the "Visit" and the "Multi-disciplinary meeting (MDM)" entities, an action is initiated (e.g. a lab test). Similarly, an association with name "preceding" indicates that a specific action serves as input to either the "Visit" or "Multi-disciplinary meeting (MDM)" entity (e.g. the result of a lab test is required for a visit to the outpatient clinic).

As indicated before, in the class diagram, we only modeled the *tests* that are selected most frequently. For each of these selected tests, the patient is required to be present. Next, for them three different ways can be distinguished in which a test is requested and ultimately the result is communicated. Note that a result is always communicated although a test might fail or that no results are obtained.

One possibility is that a test is requested and the outcome of the test is immediately reported ("T1"), Another possibility is that a test is requested, a preliminary result is communicated, followed by a final result ("T2") at a later time. The third alternative is that a test is requested and a preliminary result is communicated to either the requester or a nominated group of medical specialists. They in turn decide whether an amendment is needed ("T3"). A somewhat special case is "T4" which is similar to "T1". In addition to "T1", Proclet classes of this type may also request additional diagnostic tests for a patient in order to come to a decision. For example, for a pre-assessment test, the anesthetist might require that a lung function test is completed or a consultation with an internist. The act of requesting additional tests in order to come to a final decision are also modeled by the "follows_1" and "preceding_1" associations. These associations indicate that during a pre-assessment multiple tests can be triggered, i.e. the multiplicity is $0..*$, but also that results of multiple tests may be required as input for an examination, i.e. the multiplicity is $0..*$. Note that the requester only initiates an examination and might not be aware of the fact that additional tests need to be performed in order to arrive at an outcome.

A doctor might decide that *additional information* is required to reach the final diagnosis for a patient. Two different ways can be distinguished in which a request for additional information can be made and the result is delivered to the requester. These are: (1) additional information is requested and the requested information is immediately communicated ("A1"), (2) additional information is requested and a preliminary result is communicated to either the requester or a group of medical specialists. They in turn advise whether further investigation is required ("A2"). Note that the way in which additional information is requested, and the result communicated, is very similar to the way tests are requested and the result communicated.

During a *visit* to the hospital, the patient is examined either at the outpatient clinic or during a procedure under anesthetic. For a visit of the patient at the outpatient clinic (which can also be a consultation by telephone), several inputs might be required. These can be the results of preceding tests, i.e. the multiplicity attached to the "Test" class of association "previous_2" is $0..*$, or additional information that needs to be available, i.e. the multiplicity attached to the "Visit" class of association "previous_4" is $0..*$. Note that the results of tests and additional information may also be required as input to a multidisciplinary meeting. Therefore, the multiplicity attached to the "Visit" class of associations "previous_2" and "previous_4" is $0..1$. Moreover, as the status of a patient might be discussed during the MDO multi-disciplinary meeting ("MDO meeting"), the patient may be informed about the discussion afterwards, i.e. the multiplicity attached to "Visit outpatient clinic" and "MDO meeting" of association "previous_6" is $0..*$ and $0..1$, respectively.

During a visit by a patient to the hospital, a doctor might require a subsequent visit, i.e. the multiplicity of associations "follows_7" is $0..1$, or that a patient needs to be registered for one or more multidisciplinary meetings, i.e. the multiplicity attached to the "Multi-disciplinary meeting (MDM)" class of association "follows_6" is $0..*$.

Moreover, a doctor might also request additional information, i.e. the multiplicity attached to the "Additional information" class of association "follows_4" is $0..*$, or that tests are triggered for a patient, i.e. the multiplicity attached to the "Test" class of association "follows_2" is $0..*$. Note that tests and additional information may also be triggered for a patient during a multidisciplinary meeting. Therefore, the multiplicity attached to the "Visit" class for associations "follows_4" and "follows_2" is $0..1$.

Finally, there are the *multidisciplinary meetings* to discuss the status of multiple patients, to review the outcome of selected diagnostic tests, and to examine additional information that has been requested. Although for a certain meeting distinct inputs and outputs might exist, several commonalities can be identified. As inputs to a meeting, additional information ("previous_5"), tests ("previous_3"), and the outcome of other multidisciplinary meetings ("previous_8") might be required for *multiple* patients. Furthermore, as outputs, it might be necessary to request additional information ("follows_5"), order further tests for a patient ("follows_3"), or to initiate a multidisciplinary meeting ("follows_8"). This is done for multiple patients. Note that for the above mentioned associations, the reasoning for the multiplicities is similar to those for the "Visit" class.

As already indicated earlier, the dark-grey colored classes in Figure 6 correspond to concrete Proclet classes. Note that there is a strong correspondence between classes in a class diagram and Proclets carrying the same name. A class in a class diagram outlines the data a Proclet class carries with it and its relationship with other Proclets. Through the use of Object Constraint Language (OCL) expressions [40] it is possible to access data of different Proclets.

Note that the class diagram presented in Figure 6 is one possible solution to group and structure Proclets. Depending on the context in which the to-be modeled process is executed, a different (and perhaps more advanced) class diagram can be constructed (e.g. based on the HL7 Reference Information Model (RIM)) [41].

4.2. Proclets

As already indicated earlier, the dark-grey colored classes in Figure 6 correspond to concrete Proclet classes. In total 15 Proclet classes have been identified for the gynecological oncology workflow.

The 15 Proclet classes identified are connected to other Proclet classes via the port and channel concepts. Figure 7 shows a high-level view of the interconnection structure together with the cardinality and multiplicity of the ports. In total, there are 86 possible interactions between the Proclet classes which illustrates the complexity of the process.

By using Proclets the relationships between different entities can be described in their own process definition. So, it is more natural *to define processes as intertwined loosely-coupled object life-cycles.* As can be seen in Section 3, when using existing workflow languages, it is necessary to flatten this structure into a monolithic workflow model, which is potentially very difficult or even intractable in practice. By using Proclets, the second issue mentioned in Section 3 can be solved.

Of the 15 Proclet classes, we discuss the "Visit outpatient clinic", "Pathology", and "Pathology meeting" Proclet classes in detail. The other Proclet classes are discussed in detail in [42]. When discussing the Proclets, we will show how the limitations of monolithic workflows can be addressed by using a Proclets approach. Furthermore, we elaborate on the *interaction* of an individual Proclet with other Proclets, i.e. the *interface* of a Proclet. As can be seen in Figure 7, there can be many interactions between Proclets and even multiple interactions between the same Proclets. In order to show the kind of interactions between two Proclets, the following naming strategy is chosen for a port, consisting of several distinct parts: *sending_proclet.task_name_sending_proclet.*[*name*].*S/R*. "*sending_proclet*" refers to the Proclet class which sends the performative, *task _name_sending_proclet* refers to the specific task (or composite task) in the Pro-

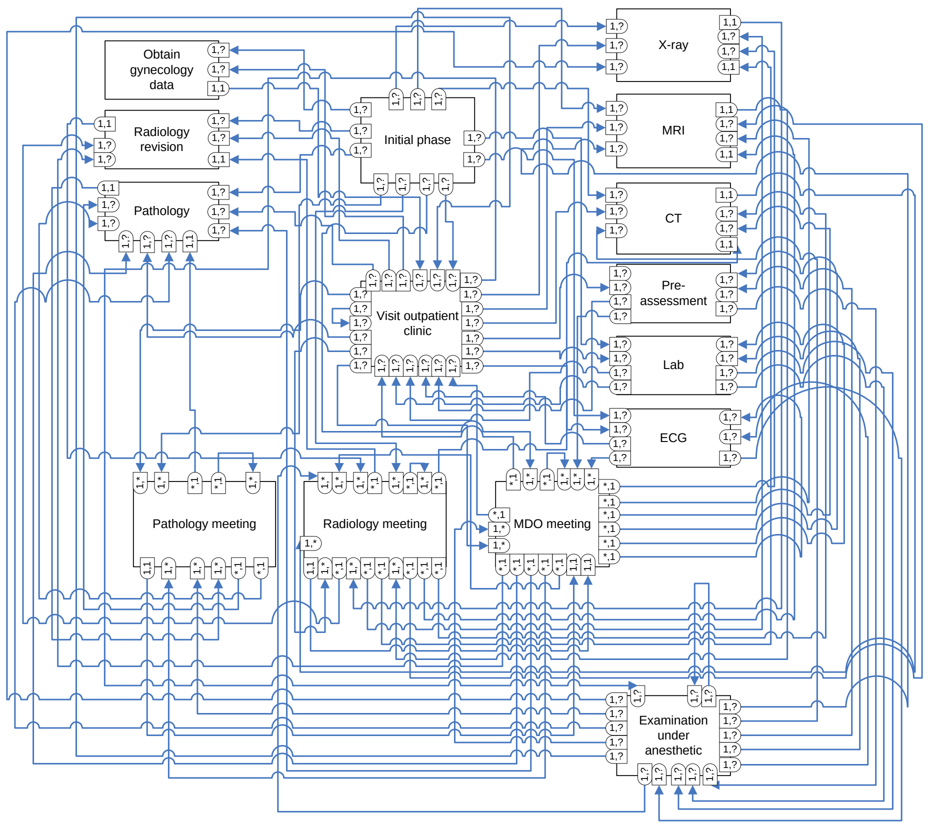


Figure 7: The Proclet classes that are defined for the healthcare process and all of the possible interactions between them.

clet class that sends the performative, "*S/R*" indicates whether a performative is sent via the port or is received via the port. "*[name]*" refers to a specific (optional) identifier that is added when the naming chosen for the other parts does not lead to a unique name. Note that by using this naming strategy each port will have a unique name. Moreover, each port can only send a performative to one other port and each port can only receive one performative from another port.

The healthcare process involves multiple medical departments, such as radiology and pathology. In order to clearly identify the resource perspective for each task in a Proclet class, it is indicated for each task which department

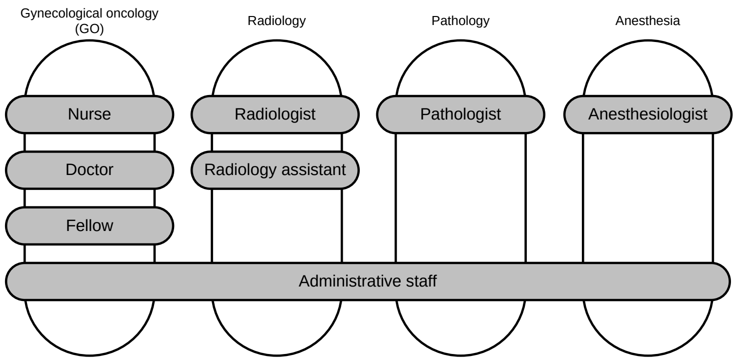


Figure 8: The organizational model for the healthcare process.

and which role is required. The corresponding organizational model is shown in Figure 8. For example, we can see that for the "gynecological oncology department", the roles "doctor" and "nurse" have been defined, and that for the "radiology" department the roles "radiologist" and "radiology assistant" have been defined.

Note that the Proclet classes discussed below are somewhat simplified in comparison to the models produced for the YAWL, FLOWer, ADEPT1, and Declare systems. First of all, the Proclet classes do not model all of the tasks that are relevant for a specific workflow. Clearly, our main motivation for defining the Proclet classes are to show the *interactions* between these Proclets as this is the core focus of the Proclet approach. Obviously, by modeling these interactions, information is included which is typically not present in a single monolithic workflow.

4.2.1. Visit Outpatient Clinic

We now analyze in detail the "Visit outpatient clinic" Proclet class that can be seen in Figure 9. This Proclet class deals with a visit by a patient to the outpatient clinic of gynecological oncology in order to see a doctor. The contents of this subprocess has already been discussed in detail in Section 3.

A visit of a patient can be requested at different parts of the process consequently triggering the creation of the respective Proclet. This is indicated by

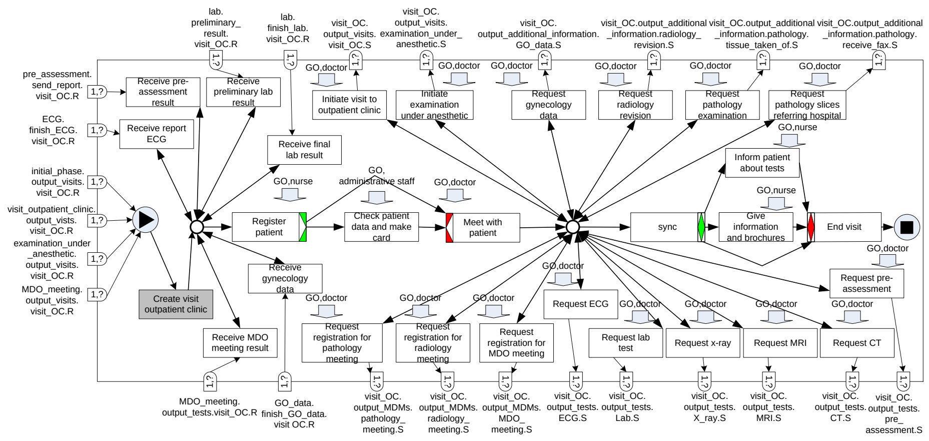


Figure 9: The “Visit outpatient clinic” Proclet class.

the cardinality 1 and multiplicity ? of the ports connected to the input condition. For example, a visit is requested during the initial stages of the healthcare process and also during a visit itself or during the MDO meeting. The next few tasks in the Proclet class deal with the meeting of the patient with the doctor (“Meet with patient” task). Directly related to such a meeting is the fact that the results of multiple tests (“Receive preliminary lab result”, “Receive final lab result”, “Receive report ECG” tasks, “Receive pre-assessment result” tasks), additional information (“Receive gynecology data” task), and the result of a MDO meeting (“Receive MDO meeting result” task) might be required as inputs. The fact that only a selection of them might be required is indicated by the cardinality 1 and multiplicity ? of the associated ports. For example, as input, the outcome of an MRI and lab test might be necessary along with the data received from the referring hospital.

Note that the tasks, required for the receipt of all the necessary inputs for a patient meeting, are modeled using a loop. Each performative received is stored in a knowledge base. The Proclet continuously inspects this knowledge base and continues with the next step (“Register patient” task) if all required performatives have been received.

During a visit to the doctor, it may be decided that several subsequent steps need to be taken in order to diagnose the patient. In general, a doctor can request that additional information is required ("Request gynecology data", "Request radiology revision", "Request pathology examination", "Request pathology slices referring hospital" tasks), that tests need to be undergone by a patient ("Request ECG", "Request lab test", "Request x-ray", "Request MRI", "Request CT", "Request pre-assessment" tasks), and that the patient needs to be discussed during a multidisciplinary meeting ("Request registration for pathology meeting", "Request registration for radiology meeting", "Request registration for MDO meeting" tasks). Moreover, a subsequent visit by the patient might be necessary ("Initiate visit to outpatient clinic", "Initiate examination under anesthetic" tasks). A doctor makes a selection of each of these steps as necessary. So, either a step is selected once or not at all. This is also indicated by cardinality 1 and multiplicity ? of the associated ports. Note that during the selection of the subsequent steps, it can be that a given Proclet needs to receive information about the different inputs that need to be received by the respective Proclet. For example, during a visit of the patient to the outpatient clinic, a doctor may decide that for the next visit of the patient an MRI and X-ray are required. In this way, when creating a new instance of a "Visit Outpatient Clinic" Proclet the content of the performative needs to indicate that for the next visit the result of both an MRI and X-ray is required.

Note that in this Proclet, *the communication with other Proclets is made explicit, i.e. communication is a first-class citizen.* In comparison to Figure 4, interaction with other processes is possible. For example, after the meeting with the doctor, subsequent steps can immediately be triggered (e.g. a lab test), whereas in Figure 4, the subprocess first needs to finish. Furthermore, in Figure 4, *subprocess dependencies are hidden in the data perspective.* For example, in Figure 4 it is not visible that during the performance of task "Meet with patient", data fields are set, which after the completion of the subprocess, cause any subsequent subprocesses to be triggered. In this way, by using Proclets, the fourth issue mentioned in Section 3 can be resolved.

Note that at run-time information held by Proclets might need to be updated or may need to be canceled. For example, as input to a meeting with a doctor, the result of a lab test might be necessary. However, the result of the lab test may not be available at the moment the meeting should take place. An option is to either cancel the whole Proclet involving the lab test or to "relink" the result of the lab test to the next meeting with the patient. At the moment, the models do not cater for the fact that Proclets can be updated or even canceled.

4.2.2. Pathology

The "Pathology" Proclet class, shown in Figure 10, describes the process in which (1) patient tissue needs to be investigated by a pathologist, (2) a request is made to review pathology material from another hospital, or (3) a request is made to reinvestigate pathology material. Analysis shows that a Proclet class can be used to *represent a workflow process which can handle multiple types of cases*. The resulting model reuses as much of the existing process as possible, i.e. no duplication of process parts is necessary or creation of separate Proclet classes.

For each type of case, specific ports are connected to the input condition in the net. For example, ports with names ending "tissue_taken_of" are used to create a Proclet instance where patient tissue needs to be investigated. Directly connected to the input condition is the light-grey colored, automatic, "Additional investigation / receive fax / tissue taken of" task. Depending on the performative received, the correct path is taken for each type of case. Note that as a different path needs to be taken on the basis of the performative received, this illustrates the need for a knowledge base.

In the situation where a request is raised asking another hospital to send its pathology material for investigation, the "Send fax" task is performed which requires a fax to be sent. After this, either the material is received ("Receive material" task) or the other hospital needs to be reminded to send the relevant material to the AMC ("Reminder" task). When the material is received, it can be investigated by a pathologist ("Investigate" task) who prepares a report ("Make report" task) that is discussed during the relevant pathology

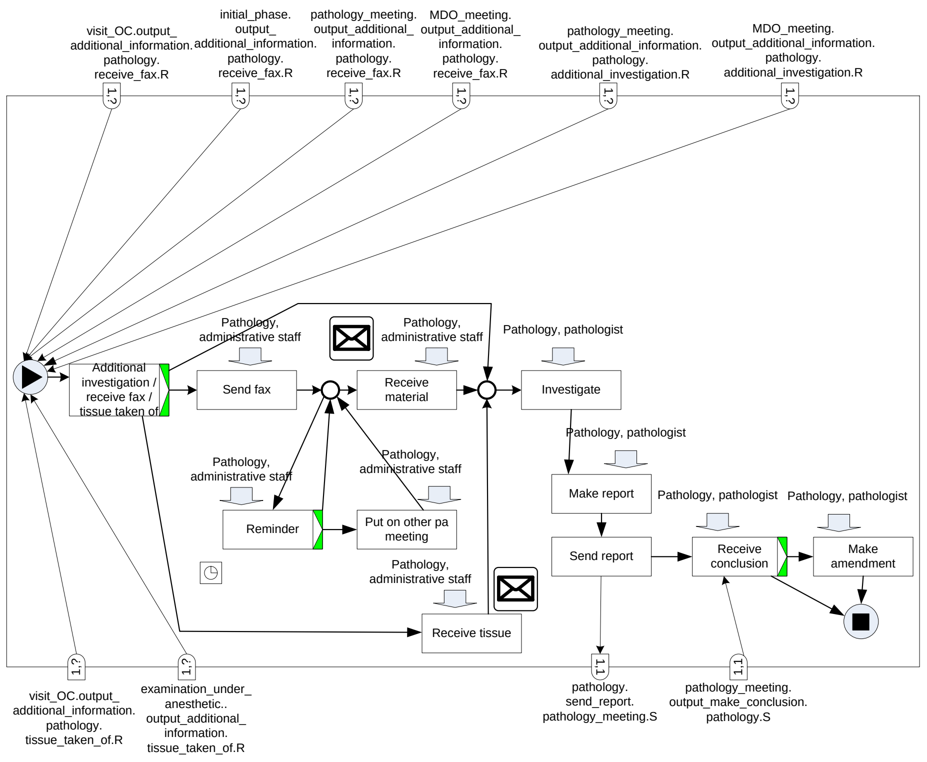

visit_OC.output_ additional_information. pathology. receive_fax.R
initial_phase. output_ additional_information. pathology. receive_fax.R
pathology_meeting. output_additional_ information. pathology. receive_fax.R
MDO_meeting. output_additional_ information. pathology. receive_fax.R
pathology_meeting. output_additional_information. pathology. additional_investigation.R
MDO_meeting. output_additional_information. pathology. additional_investigation.R
1.?
Pathology, administrative staff
Pathology, pathologist
Additional investigation / receive fax / tissue taken of
Send fax
Receive material
Investigate
Reminder
Put on other pa meeting
Make report
Send report
Receive conclusion
Make amendment
Receive tissue
visit_OC.output_ additional_information. pathology. tissue_taken_of.R
examination_under_ anesthetic.. output_additional_ information. tissue_taken_of.R
1.1
pathology. send_report. pathology_meeting.S
pathology_meeting. output_make_conclusion. pathology.S

Figure 10: The "Pathology" Proclet class.

meeting ("Send report" task). After discussion at this meeting, a performative is returned ("receive conclusion" task) which indicates whether any final amendments need to be made ("Make amendment") or not.

An alternate situation involves a request where a tissue sample taken at the AMC needs to be investigated. After receipt of the tissue ("Receive tissue"), the process follows the same course as in the previous situation, starting from the "Investigate" task.

Finally, another possible alternative involves a request where existing samples need to be reinvestigated, e.g. additional colorings might be necessary. After the request is received, the same steps are taken as for the two previous situations, starting from the "Investigate" task.

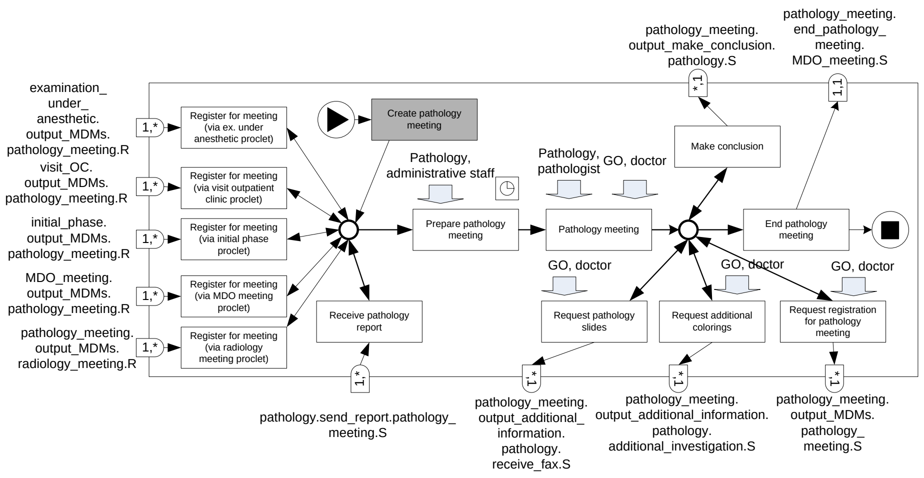

pathology_meeting. output_make_conclusion. pathology.S
pathology_meeting. end_pathology_ meeting. MDO_meeting.S
*,1
1,1
examination_ under_ anesthetic. output_MDMs. pathology_meeting.R
visit_OC. output_MDMs. pathology_meeting.R
initial_phase. output_MDMs. pathology_meeting.R
MDO_meeting. output_MDMs. pathology_meeting.R
pathology_meeting. output_MDMs. radiology_meeting.R
1,*
Register for meeting (via ex. under anesthetic proclet)
Register for meeting (via visit outpatient clinic proclet)
Register for meeting (via initial phase proclet)
Register for meeting (via MDO meeting proclet)
Register for meeting (via radiology meeting proclet)
Create pathology meeting
Pathology, administrative staff
Prepare pathology meeting
Pathology, pathologist
GO, doctor
Pathology meeting
Make conclusion
End pathology meeting
Receive pathology report
Request pathology slides
Request additional colorings
Request registration for pathology meeting
pathology.send_report.pathology_ meeting.S
pathology_meeting. output_additional_ information. pathology. receive_fax.S
pathology_meeting. output_additional_information. pathology. additional_investigation.S
pathology_meeting. output_MDMs. pathology_ meeting.S

Figure 11: The "pathology meeting" Proclet class.

4.2.3. Pathology Meeting

The "Pathology meeting" Proclet class, shown in Figure 11, describes the weekly meeting in which the gynecological oncology doctors and a pathologist discuss the samples that have been examined by a pathologist and need further review. During this meeting, samples from multiple patients are discussed. For each weekly meeting, a separate Proclet is created ("Create pathology meeting" task). In order to discuss a patient sample, it first needs to be registered (tasks starting with "Register for meeting"). This can be done at different points in the process which explains why there are multiple tasks starting with "Register for meeting" each connected to a single port. However, as is indicated by the cardinality 1 and multiplicity * of the associated ports, multiple patients can be registered using the same port. Furthermore, as a consequence of the registration for the "Pathology meeting" Proclet, a separate instance of the "Pathology" Proclet is created so that the sample can be investigated by a pathologist. The result of this Proclet is received via the "Receive pathology report" task. Consequently, there exists a tight coupling between one "Pathology meeting" Proclet and multiple "Pathology" Proclets.

Note that both the tasks for the registration for the meeting and the corresponding receipt of the results are modeled using a loop. By using a loop, multiple performatives can be received, one at a time, as indicated by cardinality 1 and multiplicity * of the associated ports. The “Pathology meeting” Proclet constantly inspects this knowledge base in order to decide whether all necessary performatives have been received and the process may continue, i.e. the “Prepare pathology meeting” task may be performed in which a pathologist prepares themselves for the pathology meeting that is held afterwards (“Pathology meeting” task).

During this meeting, it might be decided that several subsequent steps need to be taken. For each of the samples that are discussed during the pathology meeting, the corresponding “Pathology” Proclet needs to be informed whether an amendment is required or not. This involves sending a performative to each corresponding “Pathology” Proclet (“Make diagnosis”) as is indicated by the cardinality * of the accompanying port. Moreover, new pathology examinations might be required (“Request Pathology slides”, “Request additional colorings” tasks) which subsequently need to be discussed at a later pathology meeting (“Request registration for pathology meeting” task). Here also performatives need to be sent to multiple Proclets as indicated by the cardinality * of the accompanying ports. Finally, before destroying the instance, the results of the discussion are transferred to the MDO meeting so that the patient can be discussed during this meeting (“End pathology meeting” task).

As becomes clear from this analysis, *in this Proclet class we are dealing with a different level of aggregation (i.e. a group of patients) than the previous Proclet classes, and performatives are received from Proclet classes which are at a lower level of aggregation (a specific service delivered for a patient).* Using Proclets we can easily handle these differences in the level of aggregation whereas most workflow management systems force one to depict the process at an arbitrarily chosen level and to describe them in terms of one monolithic workflow. Obviously, issue number one, mentioned in Section 3, can be solved using Proclets.

Moreover, for this Proclet several *one-to-many* relationships exist within the

"Pathology" Proclet. For example, using task "Request additional colorings", it can be requested that existing samples be reinvestigated for multiple patients. The output port of this task has cardinality $*$, indicating that the performative is sent to potentially multiple recipients. In the class diagram of Figure 6, this relationship is also indicated by the "follows_5" association. Clearly, using Proclets *it can easily be captured that there are one-to-many and many-to-many relationships between entities in a workflow, whereas this is impossible to capture using existing workflow approaches.* So, issue number three, identified in Section 3, can be addressed.

5. Related Work

The first WfMSs were developed in the early 1970's. See for example the OfficeTalk system of Skip Ellis [43], an office automation system based on Petri Nets. However, only in the late nineties did these systems became more mature and more widely used in practice. Currently, there are several hundred WfMSs and workflow technology has become an integral part of numerous products, including Enterprise Resource Planning (ERP) (e.g. SAP/R3), Product Data Management (PDM), and Customer Relationship Management (CRM) systems.

Currently, administrative processes, which tend to be rather rigid, can be well supported by WfMSs. However, as indicated in [26, 44], so called "careflow systems", systems used for supporting care processes in hospitals, have special demands with regard to workflow technology. Successful implementations of workflow systems in healthcare do exist [1, 16, 25, 45, 46], for example, [25] describes how workflow technology can support the execution of recurrent tasks in a medical department. Furthermore, Zai et al. [46] recently reported on a combined workflow-informatics system for diabetes management providing each healthcare team member with just in time knowledge so that they could perform their specific tasks.

Currently, the focus of research in the field of healthcare is shifting towards describing and modeling workflow in order to design healthcare technology that provides improved support for the management of diseases. Nevertheless, the healthcare domain is still behind in adopting WfMSs for tuning healthcare organizational processes. In that way, “widespread” adoption and dissemination of workflow technology is the exception rather than the rule [27]. One of the problems that must be dealt with in order to effectively support healthcare processes using WfMSs is that more process flexibility needs to be provided by the system [47, 48]. Unfortunately, current workflow systems fall short in this area, an observation often reported in the literature [49–52].

One of the problems related to flexibility is that for a given patient a number of concurrent processes need to run in conjunction with each other [1, 53]. When complex care needs to be delivered, co-operation between various clinicians across different medical specialties and departments is needed [54], e.g. by having multidisciplinary meeting in which doctors from several medical disciplines discuss the status of a patient. In this paper, these processes can be characterized by *weakly-connected interacting lightweight workflows* in which communication and collaboration between instances of these processes is of particular importance.

Contemporary languages and systems provide limited support for the execution of lightweight interacting workflows. Instead, one is forced to squeeze real-life processes into a single “monolithic overarching workflow” which describes how an individual case is handled in isolation. In doing this, the modeler looses the overview, the natural structure of work is lost, and the required flexibility cannot be offered to the medical professionals. This issue has been recognized in literature [28, 29, 55–57] and is not limited to the healthcare domain. It also applies in other areas, e.g. the automotive domain [57], or when reviewing papers for a conference [29].

When applied to lightweight interacting workflows, the Proclet approach supports the following considerations: (1) different processes interacting with each other, (2) processes operating at different levels of aggregation, and (3) batch-oriented tasks. There are a limited range of alternate approaches to deal with some of these issues [56].

The Corepro framework [57, 58] allows for automatic generation and coordination of individual processes, operating at different levels of aggregation, based on their underlying data structure. The initial number of process instances created is decided at run-time. However, the creation of new instances at runtime is possible, but requires an ad-hoc change to the related data structures. For the Proclet approach, the number of process instances created is based on post-conditions, evaluated at runtime.

In [59–61], a two-tier, goal-driven model for workflow processes in the healthcare domain is presented. A goal-ontology, presented as a directed acyclic graph, is utilized to represent the business model at the upper level and is decomposed into an extended Petri-net model for the lower level workflow schema. A mapping is defined from the goal-graph to (sub)processes and activities such that each of the (sub)processes is designed in a way that achieves one of the upper level goals. This approach leads to a hierarchy of process models with a number of the top-level goals being implemented through subprocesses. However, there is no interaction between subprocesses in contrast to the classical way in which hierarchical processes communicate with each other in a top-down fashion.

In [55, 62, 63], the concept of artifacts is used to distinguish different levels of aggregation. However, the content of one or more artifacts can only be changed by the execution of an activity. Consequently, aggregation issues are only addressed at the activity level instead of at the process level.

Batch-oriented tasks are tasks that are based on groupings of lower aggregation elements. The concept of a batch-oriented task was already introduced in [64] in order to allow for a task that is executed for multiple instances at the same time. In [65], the problem is defined and deliberations are provided on the technology support required to deal with the issue.

Finally, in [66, 67], worklets are presented which can be seen as micro workflows. Specific activities in a process are linked to a repertoire of possible actions. Based on the properties of the case and other context information, the required action is chosen. The selection process is based on a set of rules. During enactment it is also possible to add new actions to the repertoire. However, in contrast to the Proclets Framework, these actions can only be enacted at specific parts of the process which needs to be decided at design-time. Moreover, no interactions between these worklets are possible.

6. Discussion

In this paper, we have studied the gynecological oncology healthcare process and how it can be modeled more effectively by using the Proclet framework. Due to the increased emphasis on interaction-related aspects of workflows, it is possible to model interactions between processes which otherwise would have been hidden in application logic or not been taken into account at all. At run-time, once it is known that interactions have to take place, this can be monitored leading to an improved quality of the processes that are ultimately executed. In the healthcare domain where resources are scarce and limited and often work is handed over from one resource to another, this is of the utmost importance. Currently it might be the case that a participant in a certain process is not aware of the fact that a participant in another process is waiting for input. For example, a pathologist does not start working on a report which is needed by a medical specialist a day later, but rather is working on a report which is needed two weeks later.

For the gynecological oncology healthcare process, once supported by the Proclet framework, several benefits can potentially be realized. Currently, for the radiology and pathology meetings, patients that need to be discussed must be registered for the respective meeting. However, in practice, for both meetings it happens that around 10% of the patients can not be discussed because preparatory work in the corresponding radiology and pathology processes has not taken place. Moreover, for scheduled meetings where a patient sees a doctor it is sometimes the case that a doctor finds out during the actual appointment that some results from required diagnostic tests are missing. Consequently, this leads to inefficiencies as a new appointment needs to be scheduled. In the Proclet models defined for the gynecological oncology healthcare process these interactions have been captured. Once these are monitored, these issues can be avoided.

For the Proclet classes described, there is also a link with our calendar-based scheduling support described in [68]. Today's WfMSs offer work-items to users through specific work-lists in which participants select the work-items they will perform. However, no calendar-based scheduling is offered for these work-items. In [68] it is investigated how a WfMS can be augmented with scheduling facilities such that appointments can be scheduled for these kinds of tasks. Moreover, these appointments are scheduled in the calendars of the participants involved in the actual performance of the task in order to ensure that they occur at a precise pre-agreed time suitable for all of the participants involved. Clearly, in several Proclet classes, we can also find tasks for which a concrete appointment involving specific resources needs to be made. For example, the "Meet with patient" task in the "Visit outpatient clinic" Proclet class is a task which needs to be scheduled for both a doctor and a patient. Furthermore, the "Pathology meeting" task in the "Pathology meeting" Proclet class is also a task which needs to be scheduled as a pathologist and several gynecological oncology doctors need to be available at the same time. When linking the Proclets framework with the work described in [68], also these scheduling aspects can be taken into consideration leading to better scheduling and organization of resources.

Although the Proclets framework offers various advantages, several challenges in regard to the design and implementation of the Proclet framework still remain. These challenges are discussed below.

In this paper, we only focus on the support of organizational processes by the Proclets framework. However, as mentioned in Section 1, for healthcare processes a distinction can be made between organizational processes and medical treatment processes. For the optimal support of medical treatment processes by IT, flexibility needs to be offered by CIG languages in order to effectively model CIGs [7, 26, 69, 70]. As flexibility is offered by the Proclets framework, we believe that the framework provides benefits in the modeling and execution of CIGs as well. In addition to this, until now not many studies have been devoted to the integration of CIGs and workflow languages [7, 14]. Future work can investigate the usage of the Proclets framework for providing such an integrated solution. In this way, optimal support for healthcare processes can be provided.

A limitation of our approach is that we only refer to a specific healthcare process from the gynecological oncology department within one institution, the AMC, as a test bed for the Proclets. Other healthcare processes, in particular those concerning the management of chronically ill patients, may differ in their complexity, include many more diagnostic or treatment subprocesses and more healthcare team members as part of the patient care process over a long period of time. However we consider the gynecological oncology healthcare process to be representative of other healthcare processes as it deals with the diagnosis of patients suffering from cancer, an area in which the AMC serves as a reference center. As such, we are dealing with complex diagnosis and treatment processes where multiple (sub)processes may run in conjunction with each other.

Another related aspect is that within the Proclets Framework, interaction is considered to be a first-class citizen. Consequently, when discussing the advantages, our main focus is on interaction-related aspects rather than on organizational aspects. Nevertheless, it should be noted that in healthcare the allocation of responsibilities to resources is much more dynamic than in other fields. With this in mind, future work should ensure that the dynamic and volatile nature of roles and the organization in these processes is fully taken into account. A significant achievement in this context is the work done on the so-called workflow resource patterns [71]. Independent from specific workflow technologies and modeling languages, the various ways in which resources are utilized in workflows are delineated and described in detail.

By promoting interaction to a first-class citizen, the Proclets framework, allows for modeling complex workflows in a more natural manner. However, this increased emphasis on the interaction side of workflows requires a different modeling approach, and, hence modelers trained in this new style of modeling.

Most existing modeling approaches require that a process is captured in terms of one or more complex models. By increasing the emphasis on interaction-related aspects of workflows, the Proclet framework supports the division of complex entangled processes into simple fragments. This can be achieved by first constructing a class diagram which models the concepts that exist within a process and their relationships (e.g. as in Figure 6). Afterwards, the different Proclet models and their interactions can be modeled based on the concepts identified in the class diagram and the relationships between them. Obviously this approach allows for a separation of concerns. In this way, we believe that less modeling errors are made.

Note that in regard to avoiding modeling errors, several methods are available for verification, given the modeling language that is used. For example, for the YAWL language, which is used for the Proclet models in this paper, dedicated techniques are available for verifying syntactic correctness [72]. However, this verification abstracts away from interactions and will not discover deadlocks due to inter-process communication. Hence, to adequately support Proclet-based verification, additional research is needed.

Another issue associated with promoting interactions to first-class citizens is that it is difficult to see how a series of Proclets unfolds over time. This might raise issues in communicating them to domain experts and in validating them. In order to alleviate this problem, we believe that one should first start with communicating the single Proclet models. There are several methods available which ease the communication of process models to domain experts and their validation. For example, in [30, 73, 74], a process model is validated by domain experts by means of animation. While the model is executed, the animation is updated. Another approach is discussed in [75] in which semi-formal models are used in order to elicit requirements for a socio-technical system to be built.

A Colored Petri Net (CPN) [76] model of the gynecological oncology healthcare process has been validated by means of animation as discussed in [30]. Based on these experiences we believe that animations are beneficial in communicating a process model to domain experts. Therefore, we propose to communicate Proclet models in a similar fashion which is perhaps even easier because Proclet models are small process fragments.

Once the content of the Proclet models is clear to the domain experts, as a second step the possible interactions between these models need to be communicated. For these interactions, we also propose the use animations for communicating these to domain experts. By splitting up the process of communicating the Proclet models and their interactions into two explicit steps, a separation of concerns is achieved and information overload is avoided. However, future work is needed to discover how the possible interactions between Proclet models can best be communicated or validated.

With regard to the execution of Proclets, it should be noted that currently there does not exist an engine for the enactment of Proclets. However, given the fact that Proclets have a clear semantics, such an engine can be designed and built straightaway. A related issue in this context is that for the diagnosis and treatment of a patient it can not be decided beforehand which Proclets and how many will need to be instantiated for the patient and the way in which they will interact with each other. This only becomes clear at run-time. In this way, for Proclets to find each other, this requires the definition of complex knowledge bases for them which is a far from trivial task. Next to building an engine for the enactment of Proclets, future work needs to find more natural ways to “connect” Proclets at run-time. It is important that a Proclet is aware of performatives that it will receive. If these can not be handled efficiently, escalation actions need to be taken (e.g. continue without waiting for the performative that needs to be received or cancelation of a Proclet).

Finally, with regard to the successful usage of a WfMS in healthcare it is essential that the medical professionals can enter for every task defined in a process model, the inputs required for that task. To date, WfMSs provide a wealth of advanced features for integration with existing software systems. So, the tasks that need to be done by a WfMS can well be integrated with the applications that medical professionals already use in their usual work. By integrating a WfMS in this way into the current work practices of medical practitioners, there is the advantage that the acceptance of the system is increased.

Moreover, several other interesting issues have been identified. In the gynecological oncology healthcare process many different tests can be needed by patients. These tests can be instantiated via different Proclets and the results can be received by different Proclets. Clearly, for each distinct point that such a test can be created or the result may be received a specific task with an outgoing or incoming port is required. In the future new diagnostic tests might become available which need to be included in the respective Proclets. Therefore, we see an interesting link with the worklet approach described in the related work section. Using the worklet approach, it may be possible to invoke tasks with input or output ports dynamically in Proclets instead of needing to model them explicitly.

Today's information systems record many of the events relevant to a business process. Obviously, there is an abundance of event data, and new and powerful techniques such as process mining [77, 78] provide many opportunities to analyze and improve business processes. In this context, for all process instances that are related to a given case, events are logged. Future work should focus on collecting and correlating information about tasks that were executed for different process instances, but between which a certain relationship exists.

7. Conclusion

In this paper, we have studied the gynecological oncology healthcare process and showed how it can be modeled using existing workflow languages, such as YAWL, FLOWer, Declare, and ADEPT1. Moreover, we have examined how the same process can be modeled using the Proclet framework. If we compare the Proclet framework with existing workflow languages, the following differences can be observed.

- Real-life healthcare care processes are often fragmented and composed of separate but inter-twined life-cycles running at different speeds. These processes can be effectively described using Proclets, with interaction considered as a first-class citizen, instead of straightjacketing them into one monolithic workflow.

- Real-life healthcare processes often operate at different levels of granularity. Existing workflow languages cannot take these differences into account, however, by using Proclets this is easily supported. Moreover, one-to-many and many-to-many relationships that exist between entities in a workflow, can be captured.

In healthcare, for a given patient, a lot of concurrent organizational processes can run in conjunction with each other. These processes can be created at any point in time and can also be terminated at any point in time. Moreover, these processes interact in different ways with each other. We have characterized these kinds of processes as *weakly-connected interacting lightweight workflows.* Steps in such a process may either operate at the level of a single patient or at the level of a group of patients. In other words, these processes may rely on information that resides at different levels of aggregation.

In order to successfully apply Proclets in the healthcare domain, future work related to the verification, validation, and enactment of Proclets is necessary. However, once these issues are handled, we believe that the extension of existing workflow technology with Proclet functionality can greatly assist in the modeling and enactment of care processes.

[1] R. Lenz, M. Reichert, IT Support for Healthcare Processes - Premises, Challenges, Perspectives, Data and Knowledge Engineering 61 (2007) 49–58.

[2] W. van der Aalst, K. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, Cambridge, MA, 2002.

[3] M. Dumas, W. van der Aalst, A. ter Hofstede, Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley & Sons, 2005.

[4] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag, Berlin, 2007.

[5] A. ter Hofstede, W. van der Aalst, M. Adams, N. Russell (Eds.), Modern Business Process Automation: YAWL and its Support Environment, Springer-Verlag, 2010.

[6] P. Terenziani, A Hybrid Multi-Layered Approach to the Integration of Workflow and Clinical Guideline Approaches, in: S. Rinderle-Ma, S. Sadiq, F. Leymann (Eds.), BPM 2009 International Workshops, Ulm, Germany, September 7-10, 2009, Revised Papers, 2009.

[7] N. Mulyar, W. van der Aalst, M. Peleg, A pattern-based analysis of clinical computer-interpretable guideline modeling languages, JAMIA 14 (6) (2007) 781–787.

[8] P. de Clercq, J. Blom, H. Korsten, A. Hasman, Approaches for creating computer-interpretable guidelines that facilitate decision support, Artificial Intelligence in Medicine 31 (2004) 1–27.

[9] D. Wang, M. Peleg, S. Tu, A. Boxwala, R. Greenes, V. Patel, E. Shortliffe, Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines: a literature review of guideline representation models, International Journal of Medical Informatics 68 (2002) 59–70.

[10] K. Kaiser, S. Miksch, Modeling Treatment Processes Using Information Extraction, in: H. Yoshida, A. Jain, A. Ichalkaranje, L. Jain, N. Ichalkaranje (Eds.), Advanced Computational Intelligence Paradigms in Healthcare 1, Vol. 48 of Studies in Computational Intelligence (SCI), Springer Berlin / Heidelberg, 2007, pp. 189–224.

[11] K. Kaiser, S. Miksch, Modeling Computer-Supported Clinical Guidelines and Protocols: A Survey, Vienna University of Technology, Rep. Asgaard-TR-2005-2 (2005).

[12] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, M. Stefanelli, Comparing Computer-interpretable Guideline Models: A Case-study Approach, Journal of the American Medical Informatics Association 10 (1) (2003) 52–68.

[13] P. de Clercq, K. Kaiser, A. Hasman, Computer-based Medical Guidelines and Protocols: A Primer and Current Trends, Vol. 139 of Studies in Health Technology and Informatics, IOS Press, 2008, Ch. Computer-interpretable Guideline Formalisms, pp. 22–43.

[14] J. Fox, E. Black, I. Chronakis, R. Dunlop, V. Patkar, M. South, R. Thomson, Computer-based Medical Guidelines and Protocols: A Primer and Current Trends, Vol. 139 of Studies in Health Technology and Informatics, IOS Press, 2008, Ch. From Guidelines to Careflows: Modelling and Supporting Complex Clinical Processes, pp. 44–62.

[15] M. Field, K. L. (Eds.), Institute of Medicine. Clinical Practice Guidelines: Directions for a New Program, National Academy Press, Washington, DC, 1990.

[16] K. Unertl, M. Weinger, K. Johnson, N. Lorenzi, Describing and modelling work flow and information flow in chronic disease care, Journal of the American Medical Informatics Association 16 (2009) 826–836.

[17] W. Tierney, J. Overhage, M. Murray, L. Harris, X.-H. Zhou, G. Eckert, F. Smith, N. Nienaber, C. McDonald, F. Wolinski, Effects of computerized guidelines for managing heart disease in primary care, Journal of General Internal Medicine: A Randomized, Controlled Trial 18 (2003) 967–976.

[18] A. Montgomery, T. Fahey, T. Peters, C. Macintosh, D. Sharp, Evaluation of computer based clinical decision support system and risk chart for management of hypertension in primary care; randomized controlled trial, Britisch Medical Journal 320 (686-690).

[19] A. Ozdas, T. Speroff, L. R. Waitman, J. Ozbolt, J. Butler, R. Miller, Integrating "Best of Care" Protocols into Clinicians' Workflow via Care Provider Order Entry: Impact on quality-of-care for Acute Myocardial Infarction, Journal of the American Medical Informatics Association 13 (2006) 188–196.

[20] T. East, L. Heermann, R. B. et al., Efficacy of computerized decision support for mechanical ventilation: results of a prospective multi-center randomized trial, in: N. Lorenzi (Ed.), Proceedings AMIA Annual Symposium, 1999, pp. 251–255.

[21] N. Doherty, I. Perry, The uptake and application of work flow management systems in the UK financial services sector, Journal of Information Technology 14 (2) (1999) 149–160.

[22] J. Liu, S. Zhang, J. Hu, A case study of an inter-enterprise workflow-supported supply chain management system, Information and Management 42 (3) (2005) 441–454.

[23] H. Reijers, S. Poelmans, Re-configuring workflow management systems to facilitate a "smooth flow of work", International Journal of Cooperative Information Systems 16 (2) (2007) 155–175.

[24] H. Reijers, W. van der Aalst, The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned, International Journal of Information Management 56 (5) (2005) 457–471.

[25] M. Reichert, P. Dadam, R. Mangold, R. Kreienberg, Computer-based support of clinical work processes - concepts, technologies, and their application, Zentralbl Gynakol 122 (1) (2000) 56–70, in German.

[26] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, S. Panzarasa, Flexible Guideline-based Patient Careflow Systems, Artificial Intelligence in Medicine 22 (1) (2001) 65–80.

[27] M. Murray, Strategies for the Successful Implementation of Workflow Systems within Healthcare: A Cross Case Comparison, in: R. Sprague (Ed.), Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, CA, 2003, pp. 166–175.

[28] W. van der Aalst, P. Barthelmess, C. Ellis, J. Wainer, Workflow Modeling using Proclets, in: O. Etzion, P. Scheuermann (Eds.), 7th International Conference on Cooperative Information Systems (CoopIS 2000), Vol. 1901 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2000, pp. 198–209.

[29] W. van der Aalst, P. Barthelmess, C. Ellis, J. Wainer, Proclets: A Framework for Lightweight Interacting Workflow Processes, International Journal of Cooperative Information Systems 10 (4) (2001) 443–482.

[30] R. Mans, W. van der Aalst, N. Russell, P. Bakker, A. Moleman, K. Lassen, J. Jorgensen, Transactions on Petri Nets and Other Models of Concurrency III, Vol. 5800 of Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2009, Ch. From Requirements via Colored Workflow Nets to an Implementation in Several Workflow Systems, pp. 25–49.

[31] R. Mans, W. van der Aalst, N. Russell, P. Bakker, Flexibility Schemes for Workflow Management Systems, in: Lecture Notes in Business Information Processing, Vol. 17, 2009, pp. 361–372.

[32] W. van der Aalst, A. Hofstede, YAWL: Yet Another Workflow Language, Information Systems 30 (4) (2005) 245–275.

[33] W. van der Aalst, Verification of Workflow Nets, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, Vol. 1248 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1997, pp. 407–426.

[34] W. van der Aalst, Formalization and Verification of Event-driven Process Chains, Information and Software Technology 41 (10) (1999) 639–650.

[35] W. van der Aalst, Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management, in: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets, Vol. 3098 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004, pp. 1–65.

[36] J. Searle, Speech Acts, Cambridge University Press, Cambridge, 1969.

[37] T. Winograd, F. Flores, Understanding Computers and Cognition: A New Foundation for Design, Ablex, Norwood, 1986.

[38] V. Chaudhri, A. Farquhar, R. Fikes, P. Karp, Open knowledge base connectivity 2.0, Technical Report KSL-09-06, Stanford University KSL (1998).

[39] S. White, Introduction to BPMN, BPTrends.

[40] O. M. G. (OMG), Ocl 2.0 specification, OMG document ptc/2005-06-06 (June 2005).

[41] HL7 Reference Information Model (RIM), http://www.hl7.org/implement/standards/rim.cfm.

[42] R. Mans, N. Russell, W. van der Aalst, P. Bakker, A. Moleman, Proclets in Healthcare, BPM Center Report BPM-09-05, BPMcenter.org (2009).

[43] C. Ellis, Information Control Nets: A Mathematical Model of Office Information Flow, in: Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, ACM Press, Boulder, Colorado, 1979, pp. 225–240.

[44] S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, C. Mossa, Guideline-based Careflow Systems, Artificial Intelligence in Medicine 20 (1) (2000) 5–22.

[45] K. Kuhn, J. Warren, T.-Y. Leong (Eds.), Improving compliance to guidelines through workflow technology:implementation and results in a Stroke Unit, no. 129 in Studies in Health Technology and Informatics, 2007.

[46] A. Zai, R. Grant, G. Estey, W. Lester, C. Andrews, R. Yee, E. Mort, H. Chueh, Lessons from implementing a combined workflowinformatics system for diabetes management, Journal of the American Medical Informatics Association 15 (2008) 524–533.

[47] L. Maruster, W. van der Aalst, A. Weijters, A. van den Bosch, W. Daelemans, Automated Discovery of Workflow Models from Hospital Data, in: C. Dousson, F. Höppner, R. Quiniou (Eds.), Proceedings of the ECAI Workshop on Knowledge Discovery and Spatial Data, 2002, pp. 32–36.

[48] M. Stefanelli, Knowledge and Process Management in Health Care Organizations, Methods Inf Med 43 (2004) 525–535.

[49] W. van der Aalst, S. Jablonski, Dealing with Workflow Change: Identification of Issues and Solutions, International Journal of Computer Systems, Science, and Engineering 15 (5) (2000) 267–276.

[50] W. van der Aalst, M. Weske, D. Grünbauer, Case Handling: A New Paradigm for Business Process Support, Data and Knowledge Engineering 53 (2) (2005) 129–162.

[51] C. Ellis, K. Keddara, G. Rozenberg, Dynamic change within workflow systems, in: N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, S. Kaplan (Eds.), Proceedings of the Conference on Organizational Computing Systems, ACM SIGOIS, ACM Press, New York, Milpitas, California, 1995, pp. 10 − 21.

[52] M. Klein, C. Dellarocas, A. Bernstein (Eds.), Adaptive Workflow Systems, Special Issue of Computer Supported Cooperative Work, 2000.

[53] P. Dadam, M. Reichert, K. Kuhn, Clinical Workflows - The Killer Application for Process-oriented Information Systems?, in: W. Abramowicz, M. Orlowska (Eds.), BIS2000 - Proc. of the 4th International Conference on Business Information Systems, Springer-Verlag, Poznan, Poland, 2000, pp. 36–59.

[54] R. Mans, W. van der Aalst, N. Russell, A. Moleman, P. Bakker, M. Jaspers, Modern Business Process Automation: YAWL and its Support Environment, Springer Verlag, 20009, Ch. YAWL4Healthcare, pp. 543–566.

[55] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, J. Su, Towards Formal Analysis of Artifact-Centric Business Process Models, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), International Conference on Business Process Management (BPM 2007), Vol. 4714 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2007, pp. 288–304.

[56] V. Künzle, M. Reichert, Towards Object-aware Process Management Systems: Issues, Challenges, Benefits, in: T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, R. Ukor (Eds.), Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09), Vol. 29 of Lecture Notes in Business Information Processing, Springer-Verlag, Berlin, 2009, pp. 197–210.

[57] D. Müller, M. Reichert, J. Herbst, A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures, in: R. Meersman, Z. Tari (Eds.), Advanced Information Systems Engineering, Vol. 5074 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2008, pp. 48–63.

[58] D. Müller, M. Reichert, J. Herbst, Data-Driven Modeling and Coordination of Large Process Structures, in: Z. Bellahsène, M. Léonard (Eds.), On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, Vol. 4803 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2007, pp. 131–149.

[59] E. Browne, M. Schrefl, J. Warren, A Two Tier, Goal-Driven Workflow Model for the Healthcare Domain, in: Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003), 2003, pp. 32–39.

[60] E. Browne, M. Schrefl, J. Warren, Activity Crediting in Distributed Workflow Environments, in: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004), 2004.

[61] E. Browne, M. Schrefl, J. Warren, Goal-Focused Self-Modifying Workflow in the Healthcare Domain, in: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS-37 2004) - Track 6, IEEE Computer Society Press, 2004.

[62] K. Bhattacharya, N. Caswell, S. Kumaran, A. Nigam, F. Wu, Artifact-centered operational modeling: Lessons from customer engagements, IBM Systems Journal 46 (4) (2007) 703–721.

[63] A. Nigam, N. Caswell, Business artifacts: An approach to operational specification, IBM Systems Journal 42 (3) (2003) 428–445.

[64] P. Barthelmess, J. Wainer, Workflow systems: a few definitions and a few suggestions, in: N. Comstock, C. Ellis (Eds.), Proceedings of the Conference on Organizational Computing Systems - COOCS'95, ACM Press, Milpitas, California, 1995, pp. 138–147.

[65] S. Sadiq, M. Orlowska, W. Sadiq, K. Schulz, When workflows will not deliver: The case of contradicting work practice, in: W. Abramowicz (Ed.), Proc. BIS'05, 2005.

[66] M. Adams, A. ter Hofstede, D. Edmond, W. van der Aalst, Facilitating Flexibility and Dynamic Exception Handling in Workflows, in: O. Belo, J. Eder, O. Pastor, J. Falcao e Cunha (Eds.), Proceedings of the CAiSE'05 Forum, FEUP, Porto, Portugal, 2005, pp. 45–50.

[67] M. Adams, A. ter Hofstede, D. Edmond, W. van der Aalst, Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows, in: R. Meersman, Z. Tari (Eds.), Proceedings the 14th International Conference on Cooperative Information Systems (CoopIS'06), Vol. 4275 of Lecture Notes in Computer Science, Springer-Verlag, 2006, pp. 291–308.

[68] R. Mans, N. Russell, W. van der Aalst, A. Moleman, P. Bakker, Schedule-Aware Workflow Management Systems, in: D. Moldt (Ed.), Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE09), 2009, pp. 81–96.

[69] K. Lyng, T. Hildebrandt, R. Mukkamala, From Paper Based Clinical Practice Guidelines to Declarative Workflow Management, in: D. Ardagna, M. Mecella, J. Yang (Eds.), Business Process Management Workshops: BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers, Vol. 17 of Lecture Notes in Business Information Processing, Springer-Verlag Berlin Heidelberg, 2009, pp. 336–347.

[70] N. Mulyar, M. Pesic, W. van der Aalst, M. Peleg, Declarative and Procedural Approaches for Modelling Clinical Guidelines: Addressing Flexibility Issues, in: A. ter Hofstede, B. Benatallah, H.-Y. Paik (Eds.), Business Process Management Workshops: BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers, Vol. 4928 of Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2008, pp. 335–346.

[71] N. Russell, W. van der Aalst, A. ter Hofstede, D. Edmond, Workflow Resource Patterns: Identification, Representation and Tool Support, in: Advanced Information Systems Engineering, Vol. 3520 of Lecture Notes in Computer Science, 2005, pp. 216–232.

[72] M. Wynn, H. Verbeek, W. van der Aalst, A. ter Hofstede, Business process verification: Finally a reality!, Business Process Management Journal 15 (1) (2009) 74–92.

[73] J. Jorgensen, S. Tjell, J. Fernandes, Formal requirements modelling with executable use cases and coloured Petri nets, Innovations in Systems and Software Engineering 5 (1) (2009) 13–25.

[74] J. Jorgensen, K. Lassen, W. van der Aalst, From task descriptions via colored Petri nets towards an implementation of a new electronic patient

record workflow system, International Journal on Software Tools for Technology Transfer (STTT) 10 (1) (2008) 15–28.

[75] T. Herrmann, Handbook of Research on Socio-Technical Design and Social Networking Systems, Idea Group Publishing, 2009, Ch. Systems Design with the Socio-Technical Walkthrough., pp. 336–351.

[76] K. Jensen, L. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009.

[77] W. van der Aalst, H. Reijers, A. Weijters, B. van Dongen, A. A. de Medeiros, M. Song, H. Verbeek, Business Process Mining: An Industrial Application, Information Systems 32 (5) (2007) 713–732.

[78] A. de Medeiros, W. van der Aalst, A. Weijters, Workflow Mining: Current Status and Future Directions, in: R. Meersman, Z. Tari, D. Schmidt (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Vol. 2888 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2003, pp. 389–406.