A Pattern-based Analysis of Clinical Computer-Interpretable Guideline Modeling Languages

Nataliya Mulyar[1] MSc, Wil M.P. van der Aalst[1] PhD, Mor Peleg[2] PhD

Affiliations of the authors: (1) Eindhoven University of Technology, Eindhoven, The Netherlands; (2) Department of Management Information Systems, University of Haifa, Israel

Correspondence and reprints: Nataliya Mulyar, MSc, Eindhoven University of Technology Paviljoen J.08, NL-5600 MB Eindhoven, The Netherlands. E-mail: n.mulyar@tue.nl

tel: +31 40 2475209; fax: +31 40 2432612

Abstract

Objective: Languages used to specify computer interpretable guidelines (CIGs) differ in their approaches to addressing particular modeling challenges. The main goals of this paper are: 1) to examine the expressive power of CIG modeling languages; and 2) to define the differences, from the control-flow perspective, between process languages in workflow management systems and modeling languages used to design clinical guidelines.

Design: The pattern-based analysis was applied to guideline modeling languages Asbru, EON, GLIF, and PRO*forma*. We focused on control-flow and left other perspectives out of consideration.

Measurements: We evaluated the selected CIG modeling languages and identified their degree of support of 43 control-flow patterns. We used a set of explicitly defined evaluation criteria to determine whether each pattern is supported directly, indirectly or not at all.

Results: PROforma offers direct support for 22 of 43 patterns, Asbru 20, GLIF 17, and EON 11. All four directly support. Basic Control-flow patterns, Cancellation patterns, and some Advance Branching and Synchronization patterns. None support Multiple Instances patterns. They offer varying levels of support for Synchronizing Merge patterns and State-based patterns. Some support a few scenarios not covered by the 43 control-flow patterns.

Conclusion: CIG modeling languages are remarkably close to traditional workflow languages from the control-flow perspective, but cover many fewer workflow patterns. CIG languages offer some flexibility that supports modeling of complex decisions and provide ways for modeling some decisions not covered by workflow management systems. Workflow management systems may be suitable for clinical guideline applications.

Keywords: Workflow Patterns, Evaluation, Clinical Guidelines, Modeling languages

I. Introduction

Clinical practice guidelines and protocols are being applied in diverse areas including policy development, utilization management, education, reference, clinical decision support, conduct of clinical trials, and workflow facilitation. The main intent of clinical guidelines is to improve the quality of patient care and reduce costs. Creating computer-interpretable representations of the clinical knowledge contained in clinical guidelines is crucial for developing decision-support systems that can provide patient-specific advice at the point of care. It has been demonstrated that automated guideline-based systems can improve adherence to paper-based guidelines [1].

Although many parties have been engaged in developing languages for representing computer-interpretable guidelines (CIGs) [2,3,4,5,6,7,8,9], there is still little standardization of languages that fully support representation of the logic of guidelines that unfold over time to facilitate sharing or to enable adaptation to local practice settings [10]. Indeed, the three standards, Arden Syntax, GEM and GELLO, which have been developed in the domain of clinical decision-support do not satisfy these requirements. The Guidelines Elements Model (GEM [11], a standard of the American Society for Testing and Materials (ASTM)) is an XML-based knowledge model for guideline documents. GEM elements relate to a guideline's identity, developer, purpose, intended audience, method of development, target population, knowledge components, testing, and review plan. Although this standard includes elements for marking-up elements of clinical algorithms, the resulting markup does not support computer execution that requires automatic inference. The Arden Syntax [12], a standard of ASTM and of Health Level Seven (HL7) that has been substantially used in industry. This standard is suitable for representing individual decision rules in self-contained units called Medical Logic Modules (MLMs) which are usually implemented as event-driven alerts or reminders. It is not designed for encoding complex multi-step guidelines that unfold over

time and does not offer mechanisms for complexity management and for managing linked MLMs [13]. GELLO (Guideline Expression Language, Object-oriented, see [14], which has been recently accepted as an HL7 and ANSI standard, is an object-oriented expression language that is vendor-independent, side-effect-free, and extensible expression language that could be used for specifying and sharing decision logic and eligibility criteria, calculations, patient state definitions, conditions, and system actions. As its goal is to be an expression language, it does not support specification of entire clinical algorithms, but focuses on specifying logical expressions. GELLO is the first component of a CIG language that HL7 started to standardize in order to support a full CIG formalism. The other components that HL7 sought to standardize include, among others, a control-flow language [15].

As there is no standard CIG formalism, our paper concentrates on non-standard CIG formalisms of the type termed "Task-Network Models (TNMs)" [15,14]. TNM CIG formalisms have in common a process-flow-like model that decomposes guidelines into a network of tasks that unfold over time, but they differ from each other in their approaches to addressing particular modeling challenges. In [15,14], authors compared six guideline modeling languages: Asbru, EON, GLIF, GUIDE, PRODIGY, and PRO*forma* according to eight components that capture the structure of CIG languages (see Related Work section). In this paper, we examine the modeling languages using control-flow patterns. The control-flow patterns have been tested by evaluating a multitude of workflow systems and standards. The feedback from industry has resulted in the revision and extension of the control-flow patterns, which nowadays serve as an accepted benchmark [17]. The evaluation of CIG modeling languages is a big challenge because the terminology used in these languages is inconsistent, the semantics of the control-flow of some of the languages is incompletely and informally defined, and the approaches employed by the languages for guideline modeling are heterogeneous. CIGs represent clinical algorithms that unfold over time by specifying the ordering of tasks and

activities. The ordering of tasks in a process model is also referred to in the literature as *control-flow*, which is the perspective we focused on during the analysis. We compared the control-flow component of CIG languages by evaluating their degree of support of control-flow patterns [16,17] that are known as "workflow patterns". Although workflow patterns come from the business process modeling community, they are suitable for comparing CIG languages; while a CIG language is a computer-interpretable TNM of a *clinical* care process that realizes a *clinical/medical* goal, a workflow model is a computer-interpretable TNM of a business process that realizes a *business* objective. From the control-flow perspective, both of these types of models (languages) are TNMs and are comparable from the control-flow perspective.

Initially we intended to analyze the current versions of the same set of TNM languages as considered in [15]: Asbru, EON, GLIF, NewGuide, PRODIGY and PRO*forma*. However, we excluded from our analysis NewGuide because it is still under development and PRODIGY because it is no longer actively supported.

II. Background

This section describes the main concepts of the CIG modeling languages Asbru, EON, GLIF, and PRO*forma* and presents work related to workflow patterns.

A. Computer-Interpretable Guidelines

Table 1 illustrates terms used in the CIG modeling languages which correspond to the main workflow concepts that will be used throughout this paper. These terms include process model, case, task, parallel branching and exclusive branching and are defined in [18]. A *process model* consists of a number of tasks that have to be carried out and a set of conditions that determine the order of tasks. A *task* is a logical unit of work that is carried out as a whole. Tasks can be executed based on sequential, parallel or conditional routing. *Parallel*

branching specifies that two or more tasks are executed independently of each other. *Exclusive branching* splits a process in several branches, only one of which can be selected based on the fulfillment of a condition associated with a given branch [19]. Process models are executed for specific *cases* (e.g., a patient with high blood pressure being managed by a hypertension CIG). Each case involves a process being performed, with its current active tasks. In Appendix 1 we describe in a more detail the main concepts of CIG modeling languages by modeling a patient diagnosis scenario in Asbru/AsbruView , PRO*forma*/Tallis, EON/*Protege-2000* and GLIF/Protege-2000 respectively.

B. Workflow Patterns

The recent *Workflow Patterns initiative* [17] has taken an empirical approach to identifying the most common control constructs inherent to modeling languages adopted by workflow systems. In particular, a broad survey of modeling languages resulted in 20 workflow patterns being identified [20]. The collection of patterns was originally limited to the control-flow perspective, thus the data, organizational and application perspectives were missing. In addition, the set of control-flow patterns was not complete since the patterns were gathered non-systematically: they have been obtained as a result of an empirical analysis of the modeling facilities offered by selected workflow systems. The first shortcoming has been addressed by means of the systematic analysis of data and resource perspectives and resulted in the extension of the collection of the control-flow patterns by 40 data patterns and 43 resource patterns [21,22]. The issue of the incompleteness of the control-flow patterns has been resolved by means of the systematic analysis of the classical control-flow patterns against Workflow Pattern Specification Language [23]. Furthermore, the *originally-identified* set of the 20 control-flow patterns has been revised and extended with 23 new patterns. A comprehensive description of the full set of 43 control-flow patterns is found in [16].

The 43 patterns can be divided into several groups: basic control-flow patterns, advanced branching and synchronization patterns, structural patterns, multiple instances patterns, state-based patterns, cancellation patterns, and the 23 new patterns that will be classified outside the scope of this research. Due to the lack of space, in this paper we provide only the description of patterns that have received different ratings by the examined languages, and are therefore, the most interesting. These definitions are given in the Results section, so that the discussion of the different ways in which the CIG languages support these patterns could be easily followed.

Workflow patterns have become a standard for assessing strengths and weaknesses of process specifications. Many workflow systems and standards such as XPDL, UML, BPEL, XLANG, WSFL, BPML, and WSCI were evaluated from the perspective of the control-flow patterns, a summary of which is available at [17]. The patterns have inspired the improvement and development of 10 languages and tools [17]. Furthermore, the workflow patterns were used for selecting a workflow management system (WfMS) (i.e., a system in which workflows are defined, created and executed) and have been used in teaching [17].

III. Research Questions

The main research questions addressed by this study are: "What is the degree of support of the control-flow patterns in special-purpose languages for modeling clinical guidelines?" and "What are the differences, from the control-flow perspective, between process languages offered by workflow management systems and modeling languages used to design clinical guidelines?".

IV. Methods

In this section we describe the types of analyses that we carried out and the criteria used for evaluating the pattern support offered by the examined CIG modeling languages.

A. Analysis

We evaluated the set of CIG languages against the revised set of 43 control-flow patterns, described in detail in [16].

To compare the examined languages, we used quantitative and qualitative measures. We calculated the number of patterns supported by the examined languages directly, indirectly, and not supported at all. Furthermore, we analyzed in greater detail the differences between the languages based on the support of patterns that have received different ratings. In particular, we underlined the strengths of CIG languages that were unique in their support of particular patterns, the significance of this support to clinical guidelines, the different ways in which the considered languages support the workflow patterns, and how they differ from process modeling languages used in the business domain.

B. Evaluation criteria

For each language, we checked whether it is possible to realize the control-flow pattern with the facilities offered by the language. As a means for evaluation, we used evaluation criteria explicitly defined in [16]. These evaluation criteria specify a set of context conditions an analyzed language has to fulfill in order to support a pattern. The pattern support has been rated as *full*, *partial*, or *no support*. A pattern is fully supported (+) if the examined language fully satisfies the evaluation criteria for the pattern and provides direct support for each of them via constructs found in the language. A pattern is supported partially (+/-) if the examined language provides indirect support for all of the criteria either via extended workarounds or programmatic extensions. A pattern is not supported (-) if the examined language does not satisfy any of the criteria for direct or indirect support. To make sure that our understanding of the CIG languages abilities was correct, the developers of the four languages that we compared reviewed our paper before its submission.

V. Results

A. Comparing CIG Languages Support of Categories of Workflow Patterns

Table 2 summarizes the support of the full set of 43 patterns by the languages. The brief description of pattern categories used for the evaluation is given in Table 3. We explicitly elaborate on patterns that received different ratings by the examined languages (i.e., patterns which are supported only by a sub-set of the examined languages), which underline the weaknesses and strengths of these languages essential for understanding of the paper in Appendix 2. We provide the full set of results in an online source [24].

After analyzing how the four CIG languages support the specific workflow patterns, as summarized in Table 2, we tried to arrive at more general conclusions about the languages' support of categories of Workflow Patterns. As the results of the analysis have shown, PRO*forma* offers direct support for the largest number of patterns (22 out of 43) among the examined offerings. Asbru and GLIF offer support for 20 and 17 patterns respectively. Even fewer patterns are supported by EON (it supports only 11 patterns).

More detailed analysis of the pattern support reveals that all examined offerings directly support Basic Control-flow patterns. At least half of the Advanced Branching and Synchronization patterns, which are relatively common to business processes used in practice, are supported by all offerings. Note that the Structured Synchronizing Merge pattern is not supported by all examined offerings. While PRO*forma* supports this pattern directly, Asbru adds a time restriction to the process of synchronization to approximate the desired behavior. The semantics of the Synchronization blocks in EON and GLIF are not precise enough, i.e. they do not specify what happens to the active tasks after the Synchronization task has been executed. This is also the reason why some of the new patterns addressing variants of the Synchronization Merge are not supported by EON and GLIF.

None of the examined modeling languages have the concept of a multiple instance activity and, therefore, patterns from the Multiple Instances pattern group and new patterns related to the multiple instances activity are not supported directly.

Not all examined languages have full support for the state-based patterns. Although EON and GLIF have the notion of the patient state, they lack the notion of the process state. The only language that employed these concepts is PRO*forma*. All analyzed languages support the Cancellation patterns relatively well.

B. Unique features of the CIG languages

While evaluating the modeling languages and studying their documentation, we identified several scenarios not covered by the set of the control-flow patterns that we had used as a reference framework. In particular, a deferred multi-choice is a capability to defer the selection of multiple options by a user until the user decides that no more options will be selected (for instance, selecting several medicines from the recommended ones for the treatment of the patient). The functionality of the deferred multi-choice has been encountered in GLIF3.5/*Protege-2000*, EON/*Protege-2000* and PRO*forma*/Tallis. Another scenario is related to "forced trigger", where any internal or external event triggers the execution of a task even if the task precondition was not satisfied at the moment of triggering. The functionality of the forced trigger has been encountered in PRO*forma*/Tallis.

In addition, guideline modeling languages allow for some flexibility by offering expression languages that support modeling of complex decisions. They also provide ways for modeling decisions as argumentation rules (rule-in and rule-out) which are unique features that affect control-flow specification and are not offered by workflow management systems.

Another aspect of flexibility, offered in EON and GLIF, is the ability to specify multiple entry and exit points to a guideline. Such a feature might be useful when, due to

unpredictable changes in a patient's state, a patient has to "jump" from one state of the guideline, at which he was situated at the previous encounter, to another state that reflects his current situation (e.g., his condition deteriorated despite the use of the guideline, or due to a different guideline that was applied to him, medications were added, etc). However, such support of multiple entry points is not unique to EON and GLIF and has alternatives; similar behavior can be achieved by means of the state triggers in PRO*forma*.

VI. Discussion

Members of the computerized guidelines community have emphasized how important it is to support flexibility in guideline formalisms [15,4,25]. However, when we examined guideline modeling languages, we found only limited additional flexibility not present in business process modeling languages. The CIG languages we studied support only two new patterns not encountered in business process models. This is remarkable since one would have expected dedicated constraints allowing for more flexibility given the more dynamic nature of care processes.

Moreover, only half of the workflow patterns elicited from business process modeling languages are supported by CIG languages. An interesting question is whether the patterns that are not supported by CIG languages could be useful in the domain of clinical guidelines automation. Many of these patterns relate to flexibility of process execution. In the business processes domain, multiple threads of execution that relate to the same activity are often supported (e.g., an insurance claim with a variable number of witness statements or an order containing multiple order lines). Similar situations may arise when a clinical trial is executed for groups of patients, for example. To identify whether there is a need for CIG modeling constructs supporting multiple instances, more research has to be done addressing the nature of the clinical guidelines requirements.

Since CIG languages do not offer substantially more control-flow constructs than business process modeling languages, the medical community might rethink the use of more general formalisms and tools, which have formal foundation and have been widely tested and used in industry, for expressing control flow of guideline models. For instance, the case-handling system FLOWer [26] offers a high degree of flexibility during the execution of a case (i.e., a process instance). FLOWer is based on an "information-driven" approach and takes the process as its focal point, whereas traditional workflow management systems are based on the routing of activities from work tray to work tray, leading to inflexibility. Although FLOWer suggests which steps have to be performed according to the modeled process description, a user is able to execute any task from the given list, even to re-execute some of them. This may be very useful for clinicians who are using guideline and disagree with the advice provided by the CIG because they think that their patient's case was not considered by the developers of the CIG or that new evidence suggests other treatment option. We note that some of the CIG execution engines (e.g., GLIF's execution engine GLEE) support execution of any task that is defined in the CIG, at any point in time, if the user wishes to do so. Yet, this execution semantics is not part of the semantics defined for the GLIF language.

In addition to the set of constructs discussed in this paper, the medical community may also consider using configurable modeling constructs, found in business process formalisms [27]. A CIG developed by one organization can be locally adjusted by another organization by using configurable modeling constructs. Such configurable constructs enable specifying ahead of time what part of a model can be configured and how. For instance, a choice between various kinds of tests performed by a laboratory can be configured to a choice between a blood analysis and urine analysis that are performed by an assistant of a family doctor. This is very important, as some changes that are made locally could violate the purpose of the guideline and it is therefore important to define what changes should be permitted.

Another area that has been developing in the business process community and could benefit the CIG community (especially if it would adopt a workflow-based semantics of process models) is the area of process mining [28]. Mining logs of executed events (e.g., medication ordering, patient referrals) can be used to discover the actual workflow of patient care and how it deviates from a CIGs process model.

The results of the evaluation presented in this paper could be used to clarify language specifications. Moreover, the evaluation results can be used as a means for comparing the capabilities of the languages to express the control-flow patterns and for selecting an appropriate modeling language. For instance, medical organizations, who plan to automate their processes and improve the quality of care by employing CIGs may match the list of their requirements against the results of the pattern-based evaluation. For example, if an organization requires exclusive execution of activities in non-predefined order, then Asbru might be chosen, since no other language from the evaluated ones offers these feature (see pattern 17). If a requirement is to incorporate transient triggers (pattern 23) then the best choice would be PRO*forma*; persistent triggers (pattern 24) are also supported by GLIF. PRO*forma* is also a good choice if such requirements as synchronization of variable number of paths (pattern 37) or support of milestones (pattern 18) are important. The Milestone pattern is important for modeling medical guidelines. For example, in a cancer protocol, two treatment strategies could be used: a surgery or medication. A surgery may be performed only if medication cannot be prescribed or it does not help. Checking the state of medication affect before enabling the surgery could be done by means of the Milestone pattern. GLIF or EON could be a language of choice if flexibility in the structure of a guideline is required (they support Arbitrary Cycles pattern).

The analysis we performed and reported in this paper has several limitations. It concentrates only on the control-flow aspect of the guideline formalisms and does not take

into consideration other aspects such as data and resources. Furthermore, the evaluation has been performed on the limited set of the languages. In particular, a couple of formalisms that are recognized as standards, e.g. Arden syntax and GEM, were not included in the study. Note that Arden syntax has been excluded since it is used to model individual decisions (not guidelines that unfold over time). GEM is focused on the guideline DOCUMENT model – structuring the evidence statements and the decision variable. GEM permits to markup text as imperative recommendations or as parts of decisions tables; at the same time it misses the logic of a guideline that unfolds over time.

VI. Conclusions

From a flow-control perspective, the Asbru, EON, GLIF3.5 and PRO*forma* CIG languages are very similar to the process languages of workflow management systems, although they do not make use of many of the workflow patterns in such systems. The additional workflow patterns supported by process languages of workflow management systems may be useful for clinical guideline applications. A suitable CIG can be selected for a specific modeling and execution task on the basis of pattern-based requirements. .

VII. Acknowledgments

We would like to thank Samson Tu, Silvia Miksch, Andreas Seyfang and Paolo Ciccarese for their contribution in the tool evaluations. We also thank to Richard Thomson, Nick J. Wells and John Fox for providing an access to the Tallis tool-set and for the cross-check of the evaluation results. Furthermore, we thank to Nick Russell for the input he provided to the revision of the control-flow patterns.

Appendix 1. Main concepts of CIG modeling languages

We introduce the main concepts of CIG modeling languages by modeling the following patient diagnosis scenario in the corresponding tools.

A patient is registered at a hospital, after which he consults a doctor. The doctor directs the patient to take a blood test and a urine test. When the results of both tests become available, the doctor determines the diagnosis and defines the treatment strategy.

Figure 1 presents the scenario modeled in AsbruView [29]. AsbruView is one among several tools (Delt/A [30,31], URUZ [32,33], and CareVis [34,35]) that were developed to support authoring of guidelines in Asbru [31]. A process model in Asbru [35] is represented by means of a time-oriented skeletal plan. The root plan (marked as Plan A) is composed of a set of other plans. The plans are represented as three-dimensional objects, where the width represents the time axis, the depth represents plans on the same level of the decomposition (i.e., that are performed in parallel), and the height represents the decomposition of plans into sub-plans. Parent plans are considered to be completed when all mandatory sub-plans are completed. Enabling, completion, resumption and abortion conditions can be specified for each plan, if necessary. As the time axis shows, plans *Register patient*, *Consult with doctor*, *Test phase*, and *Define the treatment* are executed sequentially. The *Test phase* plan is a parallel plan consisting of two activities: *ask for urine test* and *ask for blood test*. In this model, we used only two types of plans: sequential (root plan) and parallel plan (Test phase plan). AsbruView permits to visualize also Any-order Plan, Unordered Plan, Cyclical Plan, and If-then-else Plan, and two types of actions: Ask and Variable Assignment.

An EON model of the patient-diagnosis scenario created in the *Protege-2000* environment is illustrated in Figure 2. *Protege-2000* is an ontology-editor and knowledge-base framework (cf. http://protege.stanford.edu). The main modeling entities in EON [36] are scenarios, action steps, branching, decisions, and synchronization [37,3]. A scenario is used

to characterize the state of a patient. There are two types of Decision steps in EON, i.e. a Case step and a Choice step, which allow exactly one path or more to be selected respectively. An Action step is used to specify a set of action specifications or a sub-guideline that are to be carried out. Branch and Synchronization steps are used to specify parallel execution.

GLIF3.5 [4] is a specification method for structured representation of guidelines. To create a model in GLIF, an ontology schema and a graph widget have to be loaded into the *Protege-2000* environment. Figure 3 visualizes the GLIF model of the patient-diagnosis scenario. In GLIF3.5, five main modeling entities are used for process modeling, i.e. an Action Step, a Branch Step, a Decision Step, a Patient-State Step, and a Synchronization Step. An Action Step is a block used to specify a set of tasks to be performed, without constraints set on the execution order. It allows sub-guidelines to be included into the model. Decision steps, combining a Case Step and a Choice Step from GLIF 3.4, are used for conditional and unconditional (user-selected) routing of the flow to one out of multiple steps. Branch and Synchronization steps are used for modeling concurrent steps and synchronization of the parallel branches respectively. A Patient-State Step is a guideline step used for describing a patient state and for specifying an entry point(s) to a guideline.

PRO*forma* [39] is a formal knowledge representation language for authoring, publishing and executing clinical guidelines. It deliberately supports a minimal set of modeling constructs: actions, compound plans, decisions, and enquiries that can be used as tasks in a task network. In addition, a keystone may be used to denote a generic task in a task network. All tasks share attributes describing goals, control flow, pre-conditions, and post-conditions. A model of the patient-diagnosis scenario, created in Tallis, is shown in Figure 4. Note that in PRO*forma*, control-flow behavior is captured by modeling constructs in combination with scheduling constraints. Scheduling constraints are visualized as arrows

connecting two tasks, meaning that the task at the tail of the arrow may become enabled only after the task at the head of the arrow has completed.

The current evaluation can be considered as complementary to several comparisons [15,40, 36,9].Wang et al. [9] have reviewed and compared formal methods for CIG specification focusing mainly on guideline representation primitives, process models and their relationship with a patient's clinical status. Tu and Musen [36] focused on the computational methods of the formalisms. De Clercq et al. took a life-cycle approach to compare five formalisms for representing guidelines and medical decision rules: Asbru, EON, GLIF, PRO*forma*, and the Arden Syntax, examining guideline representation, acquisition, verification and execution aspects [40]. Peleg et al. [15] analyzed CIGs by examining them against identified by them eight dimensions capturing the conceptual components of CIGs: organization of guideline plans, goals, model of guideline actions, decision model, expression language, data interpretation/abstractions, medical concept model, and patient information model.

Appendix 2. CIG Languages Support of Workflow Patterns

The results reported in this Appendix refer to the numbering used in Table 2. We explicitly elaborate on patterns that received different ratings by the examined languages (i.e., patterns which are supported only by a sub-set of the examined languages), which underline the weaknesses and strengths of these languages essential for understanding of the paper. We provide the full set of results in an online source [24].

Basic Control-flow Patterns

The basic control-flow patterns define elementary aspects of process control: *Sequence*, *Parallel Split*, *Synchronization*, *Exclusive Choice*, and *Simple Merge*. All basic patterns are directly supported by the examined languages. We will illustrate the support of these patterns along with the description of more complex patterns.

Advanced Control-flow Patterns

While the basic control-flow patterns select all parallel paths or just one-of a set of mutually exclusive paths, the advanced patterns allow specifying in-between behaviors, where some of the paths in a set of paths can be selected for execution and different modes of continuation are possible thereafter. As shown in Table 2, two advanced Branching and Synchronization Patterns, *Multiple Choice* and *Structured Discriminator* were supported by all of the languages while Multiple Merge was not supported by any language. The *Structured Synchronizing Merge* pattern, which received different ratings for support by the examined languages, combines the functionality of the Synchronization and Simple Merge patterns used for synchronization of parallel and exclusive paths.

Pattern 7. Structured Synchronizing Merge

Description The convergence of two or more branches (which diverged earlier in the process at a uniquely identifiable point) into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled.

PRO*forma* supports this pattern via task precondition and *antecedent tasks* property specifying tasks that must be completed or discarded before this one starts. In fact, an action block used for synchronizing multiple branches would only be executed after all incoming tasks were either completed or discarded.

Asbru supports this pattern indirectly. If the branches for merging were modeled as an if-then-else statement, the merge occurs before the next step after if-then-else is performed. If the branches were implemented as sub-plans of a certain plan (using plan-ordering parallel or

unordered), the merge occurs when the continuation condition (specified in element *wait-for*) is fulfilled. The timing of the merge can be influenced via time-annotations for each plan-activation of sub-plans, both to delay it and enforce a time limit. In other words, the duration of waiting for completion of incoming tasks by the merge has to be specified. Only the inputs which arrived before the timeout occurred will be merged.

EON does not support this pattern. The only possibility for synchronization in EON is to use the *Synchronization Step* is in the mode *wait-for-all* or *proceed-after-one*, thus giving no option for synchronizing of a variable number of branches. GLIF also does not support this pattern since it does not keep track of activated branches.

Structural Patterns

Structural patterns such as *Arbitrary cycles* and *Implicit termination* identify whether the modeling formalism has any restrictions regarding the structure of the processes. We discuss the *Arbitrary Cycles* pattern below. Implicit termination is supported by all of the languages.

Pattern 10. Arbitrary Cycles

Description The ability to represent cycles in a process model that have more than one entry or exit point.

Only GLIF and EON support this pattern. In GLIF it is possible to specify multiple entry or end-points to a loop (see Figure 5). In addition, it is possible to specify iterations of action and decision steps using an iteration expression that specifies the iteration frequency, along with stopping and abort conditions that terminate the loop. Similar structures can be realized in EON. Asbru supports only structured loops (i.e., without multiple entry and exit points) by means of a cyclical plan (see Figure 6). PRO*forma* prohibits modeling arbitrary cycles to prevent a model from deadlocking. Note, however, that all analyzed languages allow for modeling of the structured loops (also known as while-do and repeat-until constructs of pattern 21).

Multiple Instances Patterns

The Multiple Instances (MI) patterns refer to situations where several instances of a task can be active concurrently in the same case. None of the examined languages offers a direct support for these patterns; therefore we omit the discussion of patterns related to them.

State-based Patterns

The state-based patterns characterize scenarios in a process where subsequent execution is determined by the state of the process instance. There are three such patterns: *Deferred Choice*, *Interleaved Parallel Routing* and *Milestone*. We discuss the *Interleaved parallel Routing* pattern in the context of the *Critical Section* pattern (#39), which is a more complex pattern variant of it. The two other state-based patterns are described below.

Pattern 16. Deferred Choice

Description A point in a workflow process where one of several branches is chosen based on interaction with the operating environment. Prior to the decision, all branches present possible future courses of execution. After the decision is made, execution alternatives in branches other than the one selected are withdrawn.

From the analyzed specifications, Asbru, GLIF, and PRO*forma* support this pattern. Asbru implements this pattern via the any-order plan in conjunction with the continuation specification *wait-for-one* and the flag *confirmation required* in the filter-condition of the sub-plans. In this case, all sub-plans are presented to the user who selects one sub-plan. As soon as this sub-plan is activated (in response to the user's selection) the other sub-plans cannot be activated any more (because of the mechanism of any-order plan). As soon as the selected sub-plan is completed, the parent plan completes, because it was only waiting for one sub-plan to complete. Thus, the not selected plans cannot be selected later.

GLIF supports this pattern by a *Decision Step* with no conditions specified on the outgoing arcs. When multiple options are presented to a user, recommendations for selecting

or declining the presented options are given to the user. The recommendations for the decision may involve rule-in, rule-out, strict-rule-in and strict-rule-out properties. These properties contain a set of conditions which has to be satisfied in order to suggest which candidate to select and which to decline.

PRO*forma* supports this pattern by a *Decision* plan in which choice is made by an end-user between different candidates. The selection of a candidate is driven by an argument in the form of the truth-valued expression and support offered to the candidate if the condition is true. Next to this, Decision has recommendation rules which determine whether a certain candidate is recommended or not. To make sure that only one candidate from multiple available ones will be selected, a selection mode has to be set to *single*. An end-user may select either a recommended or a non-recommended candidate. The result of the *Decision* block used in preconditions of the tasks following this *Decision* realizes the behavior of the *Deferred Choice.*

Note that although in EON a *Choice Step* and the associated *Action Choice* present several choices to a user and the decision as to which option is selected is deferred until the user makes his choice, multiple options can be selected, thus the option to execute other branches is not immediately withdrawn.

Pattern 18. Milestone

Description An activity is only enabled when the process instance (or which it is part) is in a specific state. The state is assumed to be a specific execution point (also known as a *milestone*) in the process model. When this execution point is reached, the nominated activity can be enabled. If the process instance has progressed beyond this state, then the activity cannot be enabled now or at any future time (i.e. the deadline has expired).

PRO*forma* supports this pattern by means of a state trigger that allows checking states of activities and values of truth-valued expressions. Asbru does not support this pattern. EON

also does not support this pattern, however it allows to represent a state of a patient via *Scenario*, the eligibility conditions of which specify the necessary conditions for a patient to be in this scenario. GLIF does not support this pattern directly, however it allows an Action Step to be triggered by an event of the following types: end-of-previous step, patient-data-availability, patient-arrival, or temporal. Similar to EON, GLIF allows for the modeling of a *Patient State* step which denotes the state of the patient. *Patient State* step is used to denote multiple entry points to a guideline model.

Cancellation Patterns

There are two so-called cancellation patterns: *Cancel Activity* and *Cancel Case*. *Cancel Activity* pattern is supported by all examined offerings. Asbru supports canceling of an arbitrary set of tasks whereas other languages support either canceling of a single task or of a group of tasks related to each other.

Pattern 20. Cancel Case

Description A complete process instance is removed. This includes currently executing activities, those which may execute at some future time and all sub-processes.

Asbru supports this pattern via an abort-condition of a root plan. PRO*forma* supports this pattern via an abort condition associated with a plan containing all activities. GLIF indirectly supports this pattern. It requires the whole guideline to be placed inside of the *Action Step*, the fulfillment of the abort condition of which would lead to the cancellation of the included guideline. EON does not support this pattern.

New Patterns

The 23 new control-flow patterns consist of a set of completely new patterns and a number of variants of the revised patterns described earlier. Among them are patterns which address the concepts such as triggers, path and thread branching and synchronization, and cancellation. Only 11 patterns from this category are supported by some of the analyzed languages. We

omit the discussion of patterns *Cancel Multiple Instance Activity* (#26) and *Complete Multiple Instance Activity* (#27) because they relate to the concept of multiple-instances task which is not present in of the examined languages. These patterns are supported by some of the languages because they offer support for task cancellation and task completion. We discuss the support of the *Interleaved Routing* pattern (#40) by means of its complex pattern variant *Critical Section* (#39). The other 8 patterns are described below.

Pattern 22. Recursion

Description The ability of an activity to invoke itself during its execution or an ancestor in terms of the overall decomposition hierarchy with which it is associated.

EON, GLIF and PRO*forma* do not support this pattern. Asbru supports it by a static-plan pointer invoking itself in an invoking-plan element.

Pattern 23. Transient Trigger

Description The ability for an activity to be triggered by a signal from another part of the process or from the external environment. These triggers are transient in nature and are lost if not acted on immediately by the receiving activity.

From all examined specifications, only PRO*forma* supports this pattern via an event trigger, which brings a task to execution even if scheduling constraints are not met. Since the context conditions of this pattern assume that the transient triggers are lost if not acted on immediately, and PRO*forma* event triggers always force the execution of tasks and never get lost, therefore PRO*forma* supports this pattern partially.

Pattern 24. Persistent Trigger

Description The ability for an activity to be triggered by a signal from another part of the process or from the external environment. These triggers are persistent in form and are retained by the workflow until they can be acted on by the receiving activity.

GLIF supports this pattern via Action steps and Decision steps with an attribute *triggering events* (see Figure 7), which specifies the events that trigger the execution of the step. During the execution, when the flow reaches a step that has associated triggering events, this next step should be executed only after one of its triggering event occurred. If more than one triggering event occurs at the same time, then the highest priority event is chosen to trigger the step. PRO*forma* supports the pattern via a state trigger. A state trigger is an expression that has to be true before the task can be executed. The task remains dormant until it becomes true. Asbru and EON do not support this pattern.

Pattern 29. Canceling Discriminator

Description The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled. Triggering the discriminator also cancels the execution of all of other incoming branches and resets the construct.

Only PRO*forma* and Asbru support this pattern. PRO*forma* supports this pattern via a plan which has tasks that are marked as terminal. The plan completes as soon as a first terminal task has completed (see Figure 8). Asbru supports this pattern by means of the *Propagation Specification* (abort-if) to influence the *Abort-Condition*.

We describe the support of the *Structured N-out-of-M Join pattern* which automatically supports special case of the 1-out-of-M Join also known as the *Structured Discriminator* pattern (#9).

Pattern 30. Structured N-out-of-M Join

Description The convergence of M branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Subsequent

enablement of incoming branches does not result in the thread of control being passed on. The join construct resets when all active incoming branches have been enabled.

Asbru supports the pattern by *wait-for-group* attribute of a plan that specifies that *N* tasks must complete to continue the execution, the rest of the tasks are out of importance. GLIF supports the pattern via a *Synchronization* step whose continuation attribute allows for explicit specification of branches which must complete before a subsequent activity can be performed (see Figure 9). PRO*forma* supports this pattern via a plan which has tasks that are marked as terminal. Completion of all tasks in any of the specified groups would lead to termination of the discriminator and cancellation of active tasks (see Figure 10). Although EON supports 1-out-of-M join, it does not support this pattern.

Pattern 32. Canceling N-out-of-M Join

Description The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Triggering the join also cancels the execution of all of the other incoming branches and resets the construct.

None of the examined CIGs except for PRO*forma* support this pattern. PRO*forma* supports this pattern directly as described in the *Structured N-out-of-M Join* pattern.

Pattern 37. Acyclic Synchronizing Merge

Description The convergence of two or more branches which diverged earlier in the process into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled. Where a given branch does not have a thread of control passed to it at the divergence, "false" tokens are passed along the branch to ensure that the merge construct is able to determine when each of the incoming branches can

be synchronized. Clearly, it is only possible to pass false tokens if the split is before the join. Therefore, no cycles are possible.

None of the examined CIGs except PRO*forma* support this pattern. PRO*forma* supports this pattern directly via scheduling constraints and the status of the antecedent tasks.

The patterns Interleaved Parallel Routing (pattern 17), Critical Section (pattern 39), and Interleaved Routing (pattern 40) address similar problems; therefore we will describe only one of them, i.e., the Critical Section pattern.

Pattern 39. Critical Section

Description. Two or more connected subgraphs of a process model are identified as "critical sections". At runtime for a given process instance only activities in one of these "critical sections" can be active at any given time. Once execution of the activities in one "critical section" commences, it must complete before another "critical section" can commence.

Asbru supports the pattern by *Any-Order* sub-plans, where critical sections have to be included in a body of sub-plans (see Figure 11). GLIF supports this pattern via *any_order* attribute of Branch step. Critical sections have to be included on separate branches. EON and PRO*forma* do not support this pattern.

References

1. J.M. Overhage, W.M.Tierney, X.H. Zhou, C.J. McDonald. A Randomized Trial of "Corollary Orders to Prevent Errors of Omission." J Am Med Inform Assoc. 1997;4(5):364-375.

2. P. Votruba, S. Miksch, and R. Kosara. Facilitating Knowledge Maintenance of Clinical Guidelines and Protocols. Medinfo. 2004;11(Pt 1):57-61

3. S.W. Tu. The EON guideline model. Technical report. Last accessed May 2, 2007, http://smi.stanford.edu/projects/eon/EONGuidelineModelDocumentation.doc, 2006.

4. A.A. Boxwala, M.Peleg, S. Tu, O.Oqunyemi, Q. Zeng, D. Wang, and et al. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. Biomedical Informatics, 37(3):147–161, 2004.

5. P.D. Johnson, S.W. Tu, N. Booth, B. Sugden, and I.N. Purves. Design and implementation of a framework to support the development of clinical guidelines. Proc AMIA Symp., pp. 389–393, 2000.

6. P.A. de Clercq, A. Hasman, J.A. Blom, and H.H. Korsten. Design and implementation of a framework to support the development of clinical guidelines. Int J Med Info, 64(2), 2001.

7. P. Ciccarese, E. Caffi, S. Quaglini, and M. Stefanelli. Architectures and tools for innovative Health Information Systems: The Guide Project. Int J Med Info, 74(7-8), 2005.

8. S. Tu and M. Musen. A flexible Approach to Guideline Modeling. In AMIA Symp., pp. 420–424, 1999.

9. D. Wang, M. Peleg, S. Tu, A.A. Boxwala, and R.A. Greenes et al. Representation Primitives, Process Models and Patient data in Computer-interpretable Clinical Practice Guidelines: A Literature Review of Guideline Representation Models. Intl J Med Inform, 68(1), 2002.

10. M. Peleg , Chapter 13: Guideline and Workflow Models. In Greenes R.A., editor, Medical Decision Support: Computer-Based Approaches to Improving Healthcare Quality and Safety. Elsevier, 2006, pp. 281-306..

11. R.N. Shiffman, B.T. Karras, A. Agrawal, R. Chen, L. Marenco, S. Nath. GEM: a proposal for a More Comprehensive Guideline Document Model Using XML. J Am Med Inform Assoc. 2000;7(5):488-498

12. G. Hripcsak, P. Ludemann, T.A. Pryor, O.B. Wigertz , P.D. Clayton. Rationale for the Arden Syntax. Comput Biomed Res 1994;27(4):291-324

13. M. Peleg, A.A. Boxwala, E. Bernstam, S. Tu, R.A. Greenes, E.H. Shortliffe. Sharable Representation of Clinical Guidelines in GLIF: Relationship to the Arden Syntax. Journal of Biomedical Informatics 2001;34(3):170-81.

14. M. Sordo, O. Ogunyemi, A.A. Boxwala, R.A. Greenes, S. Tu. GELLO: A common expression language. http://cslxinfmtcs.csmc.edu/hl7/arden/2004-09-ATL/v3ballot_gello_aug2004.zip)

15. M. Peleg, S.W. Tu, J. Bury, P. Ciccarese, J. Fox,. R.A. Greenes and et al. Comparing Computer-interpretable Guideline Models: A Case-study Approach. J Am Med Inform Assoc., Vol. 10, No. 1, Jan-Feb 2003, pp. 52-68.

16. N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org, 2006, URL: http://workflowpatterns.com/documentation/documents/BPM-06-22.pdf, last accessed April 30, 2007.

17. Workflow Patterns Home Page. http://www.workflowpatterns.com. Last accessed April 30, 2007.

18. W. van der Aalst and K. van Hee. Workflow management. The MIT press, 2004.

19. M.Dumas, W. van der Aalst, A.H. ter Hofstede. Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley&Sons. 2005.

20. B. Kiepuszewski. Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003. Available via http://www.workflowpatterns.com.

21. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns: Identification, Representation and Tool Support, *CAISE 2005*, Porto, Portugal, pp. 216-232.

22. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.

23. N. Mulyar, W.M.P. van der Aalst, A.H.M. ter Hofstede, and N. Russell. Towards a WPSL: A Critical Analysis of the 20 Classical Workflow Control-flow Patterns. Technical report, BPM Center Report BPM-06-18, BPMcenter.org, 2006.

24. N. Mulyar, W.M.P. van der Aalst, and M. Peleg. A Pattern-based Analysis of Clinical Computer-Interpretable Guideline Modeling Languages. BPM Center Report BPM-06-29, BPMcenter.org, 2006.

25. S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso and S. Panzarasa. Flexible guideline-based patient careflow systems. Artif Intell Med. 2001 Apr;22(1):65-80.

26. Pallas Athena. Pallas Athena website. http://www.pallasathena.nl. Last accessed May 2, 2007.

27. W.M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, M. H. Jansen-Vullers: Configurable Process Models as a Basis for Reference Modeling. Business Process Management Workshops 2005, LNCS, Springer, pp.512-518, 2005.

28. A. Rozinat, R.S. Mans, and W.M.P. van der Aalst. Mining CPN Models: Discovering Process Models with Data from Event Logs. In K. Jensen, editor, Proceedings of the Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2006), Aarhus, Denmark, October 2006. University of Aarhus.

29. AsbruView. 13 December 2006 http://www.ifs.tuwien.ac.at/asgaard/asbru/tools.html.

30. DELT/A. Document Exploration and Linking Tool/Addons. http://ieg.ifs.tuwien.ac.at/projects/delta. Last accessed May 2, 2007.

31. Y. Shahar, S. Miksch, and P. Johnson. The Asgaard Project: a task-specific Framework for the application and critiquing of time-oriented clinical guidelines. Artif Intell Med, (14):29–51, 1998.

32. R. Kosara and S. Miksch. Metaphors of Movement: A Visualization and User Interface for Time-Oriented Skeletal plans. Artificial Intelligence in Medicine , Special Issue, 22(2), pp. 111-131, 2001.

33. CareVis. http://ieg.ifs.tuwien.ac.at/projects/carevis. Last accessed April 30, 2007.

34. W. Aigner and S. Miksch. CareVis: Integrated Visualization of Computerized Protocols and Temporal Patient Data. Artif Intell Med, Vol. 37, No. 3. (July 2006), pp. 203-218.

35. A. Seyfang, R. Kosara, and S. Miksch. Asbru Reference Manual, Version 7.3. Technical report, Vienna University of Technology, Institute of Soft-ware Technology, Vienna. Report No.: Asgaard-TR-2002-1, 2002.

36. S. Tu and M. Musen. Representation Formalisms and Computational Methods for Modeling Guideline-Based Patient Care. In First European Workshop on Computer-based Support for Clinical Guidelines and Protocols, Leipzig, Germany, pp.125–142, 2000.

37. S.W. Tu and M.A. Musen. A flexible approach to guideline modeling. In Proc AMIA Symp, pp. 420–424, 1999.

38. S.W. Tu and M.A. Musen. From guideline Modeling to guideline execution: Defining guideline-based decision-support Services. In Proc AMIA Annu Symp, pp. 863–867, 2000.

39. J. Fox, N. Johns, and A. Rahmanzadeh. Disseminating Medical Knowledge: The PROforma Approach. Artificial Intelligence in Medicine, 14(1):157–182, 1998.

40. P.A. Clercq, J.A. Blom, H.H. Korsten, and A. Hasman. Approaches for creating computer interpretable guidelines that facilitate decision support. Artif Intell Med. 2004 May;31(1):1-27.

Table 1 Terms used by Asbru, EON, GLIF, and PRO*forma*

Terms	**Asbru**	**EON**	**GLIF**	**PRO*forma***
Process model	Plan	Guideline	Guideline	Plan
Case	Instance of Plan	Guideline Instance	Guideline Instance	Instance of Plan
Task/ activity	Plan	Action	Action	Action, Enquiry
Parallel branching	Plan type	Branch and Synchronization	Branch and Synchronization	Action or Enquiry
Exclusive branching	Plan precondition, Plan type	Decision	Decision	Decision, Enquiry and scheduling constraints

Table 2 Support for the Control–flow Patterns in Asbru, EON, GLIF, and PRO*forma*

Basic control-flow	Asbru	EON	GLIF	PRO*forma*
1. Sequence	+	+	+	+
2. Parallel Split	+	+	+	+
3. Synchronization	+	+	+	+
4. Exclusive Choice	+	+	+	+
5. Simple Merge	+	+	+	+
Advanced Branching and Synchronization				
6. Multi-choice	+	+	+	+
7. Structured Synchronizing Merge	+/-	-	-	+
8. Multi-merge	-	-	-	-
9. Structured Discriminator	+	+	+	+
Structural Patterns				
10. Arbitrary Cycles	-	+	+	-
11. Implicit Termination	+	+	+	+
Multiple Instances Patterns				
12. MI without Synchronization	-	-	-	-
13. MI with a priori Design Time Knowledge	+/-	+/-	+/-	+/-
14. MI with a priori Run-Time Knowledge	-	-	-	-
15. MI without a priori Run-Time Knowledge	-	-	-	-
State-Based Patterns				
16. Deferred Choice	+	-	+	+
17. Interleaved Parallel Routing	+	-	-	-
18. Milestone	-	-	-	+
Cancellation Patterns				
19. Cancel Activity	+	+	+	+
20. Cancel Case	+	-	+/-	+
New patterns				
21. Structured Loop	+	+	+	+
22. Recursion	+	-	-	-
23. Transient Trigger	-	-	-	+
24. Persistent Trigger	-	-	+	+
25. Cancel Region	-	-	-	-
26. Cancel Multiple Instance Activity	+	-	+	+
27. Complete Multiple Instance Activity	+	-	-	+
28. Blocking Discriminator	-	-	-	-
29. Canceling Discriminator	+	-	-	+
30. Structured N-out-of-M Join	+	-	+	+
31. Blocking N-out-of-M Join	-	-	-	-
32. Canceling N-out-of-M Join	-	-	-	+
33. Generalized AND-Join	-	-	-	-
34. Static N-out-of-M Join for MIs	-	-	-	-
35. Static N-out-of-M Join for MIs with Cancellation	-	-	-	-
36. Dynamic N-out-of-M Join for MIs	-	-	-	-
37. Acyclic Synchronizing Merge	-	-	-	+
38. General Synchronizing Merge	-	-	-	-
39. Critical Section	+	-	+	-
40. Interleaved Routing	+	-	+	-
41. Thread Merge	-	-	-	-
42. Thread Split	-	-	-	-
43. Explicit Termination	-	-	-	-

(+) : full support; (+/-): partial support, (-) : no support. MI – Multiple Instances

Table 3 Description of pattern categories

Category name	Description
Basic control-flow patterns	Patterns describing elementary aspects of process control: *Sequence*, *Parallel Split*, *Synchronization*, *Exclusive Choice*, and *Simple Merge*
Advanced Branching and Synchronization	Patterns describing in-between behaviors, where some of the paths in a set of paths can be selected for execution and different modes of continuation are possible thereafter
Structural Patterns	Structural patterns identify whether the modeling formalism has any restrictions regarding the structure of the processes
Multiple Instances Patterns	Patterns that refer to situations where several instances of a task can be active concurrently in the same case
State-Based Patterns	Patterns characterizing scenarios in a process where subsequent execution is determined by the state of the process instance
Cancellation Patterns	Patterns refer to the situation where either a single task or a group of tasks have to be cancelled in a model
New patterns	A set of new patterns and the revised variants of patterns in the above-introduced categories which address the concepts such as triggers, path and thread branching and synchronization, and cancellation.

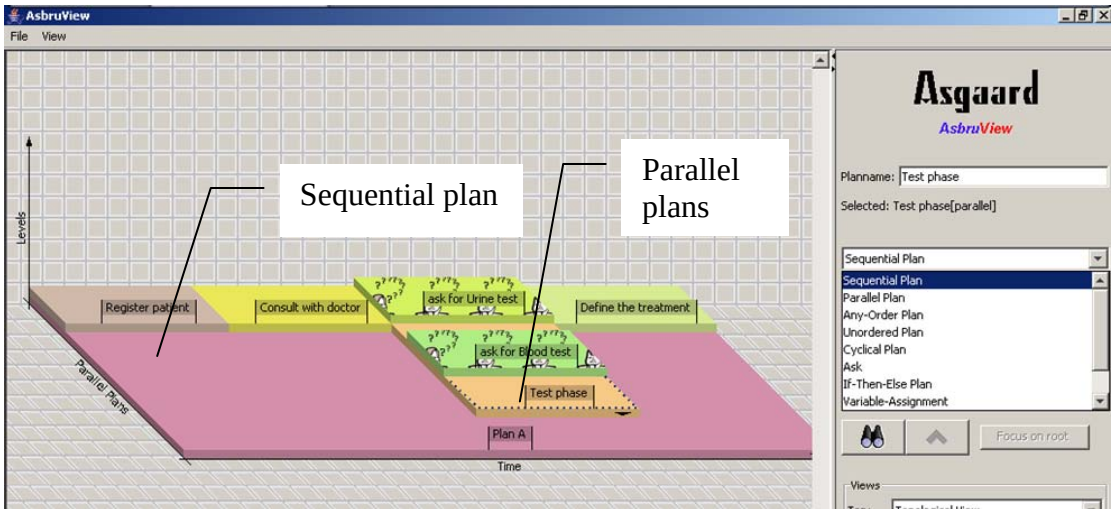

Figure 1 The patient-diagnosis scenario modeled in AsbruView

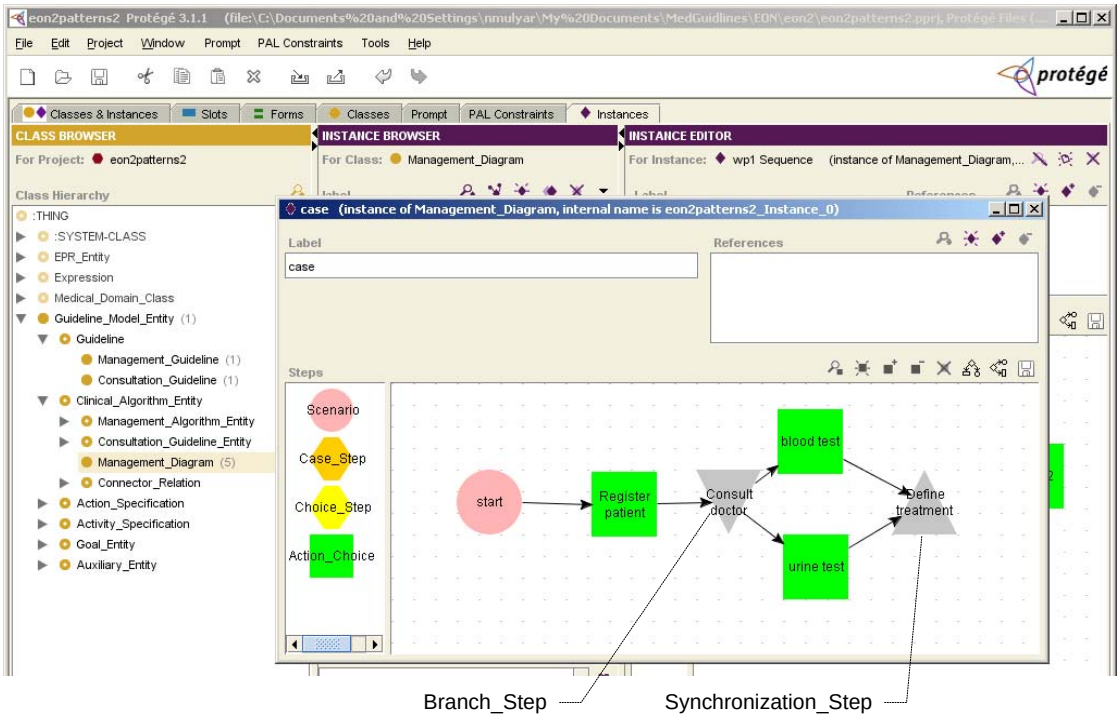

Figure 2 The patient-diagnosis scenario modeled in EON/Protege

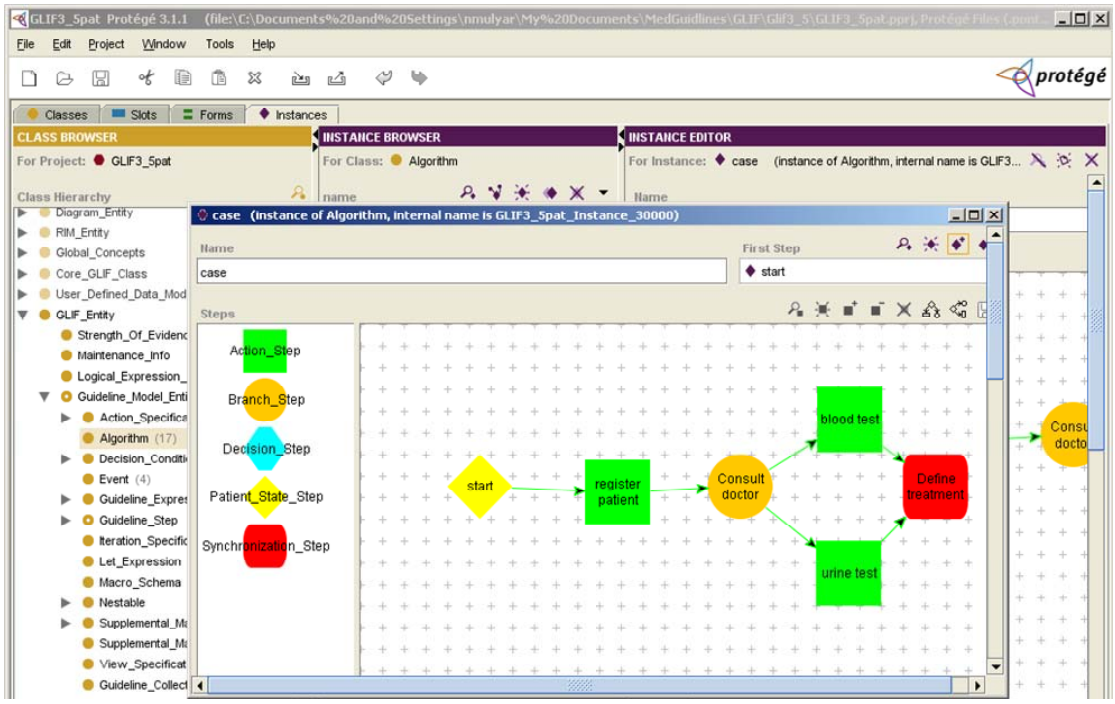

Figure 3 The patient-diagnosis scenario modeled in GLIF3.5/Protege

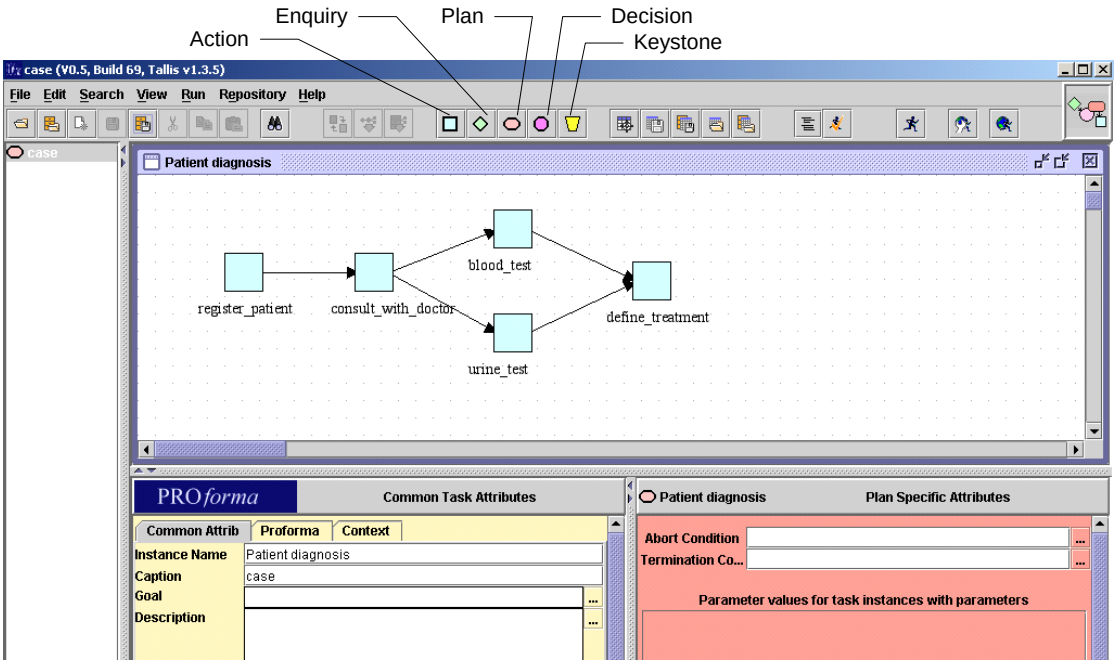

Figure 4 The patient-diagnosis scenario modeled in PRO*forma*/Tallis

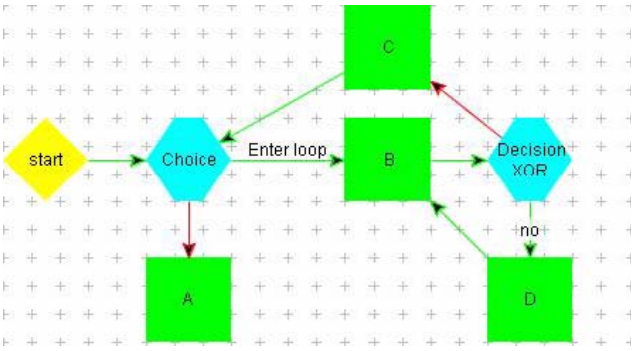

Figure 5 Specification of the Arbitrary Cycles in GLIF3.5/Protege

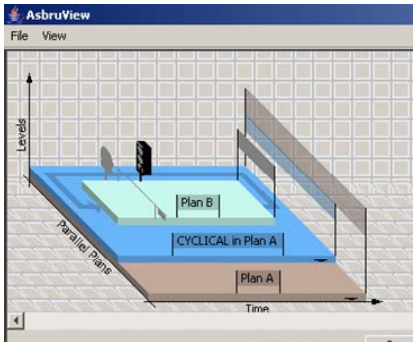

Figure 6 Specification of the Structured Loop in Asbru/AsbruView

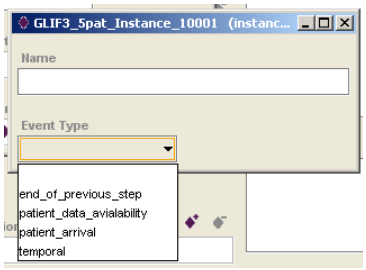

Figure 7 Specification of the Persistent Trigger in GLIF3.5

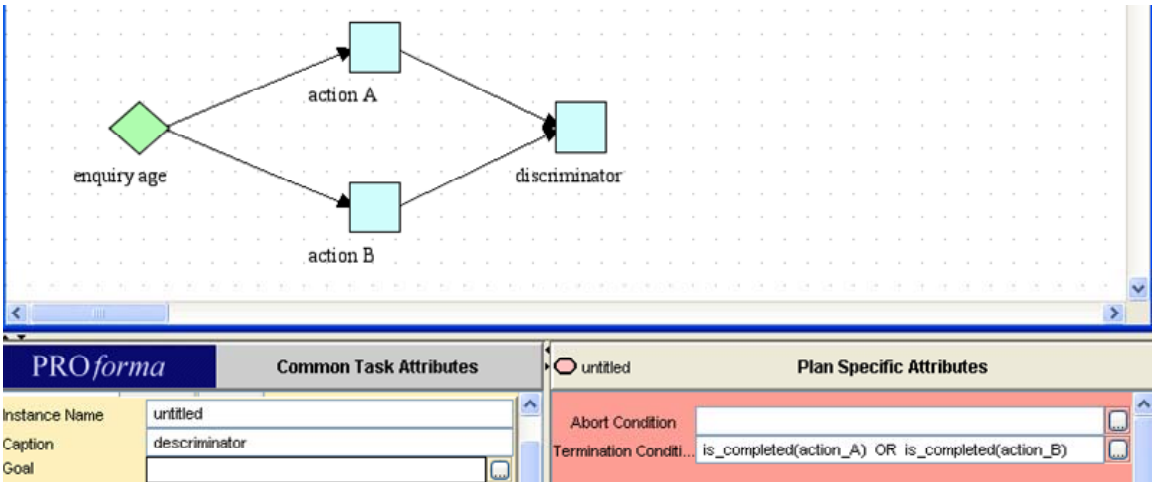

Figure 8 Specification of the Canceling Discriminator in PROforma/Tallis

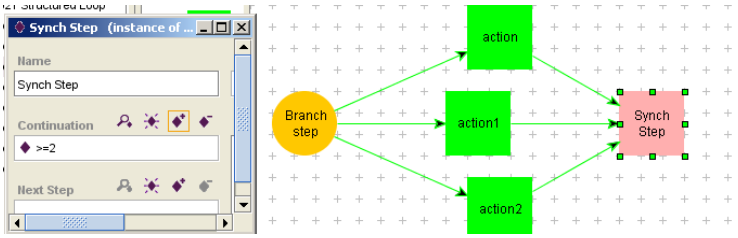

Figure 9 Specification of the Structured N-out-of-M join in GLIF3.5/Protégé

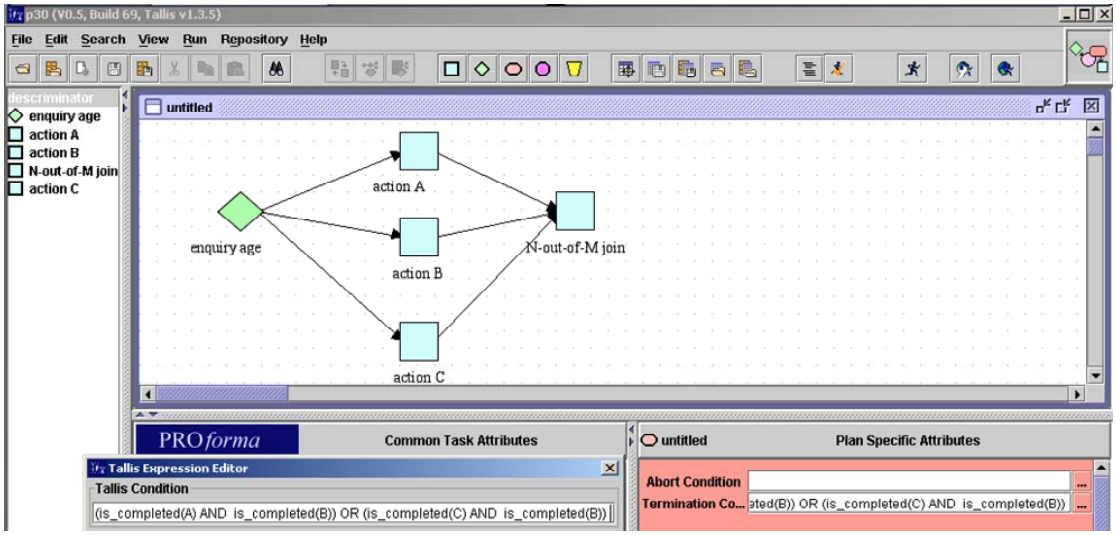

Figure 10 Specification of the Structured N-out-of-M join in PROforma/Tallis

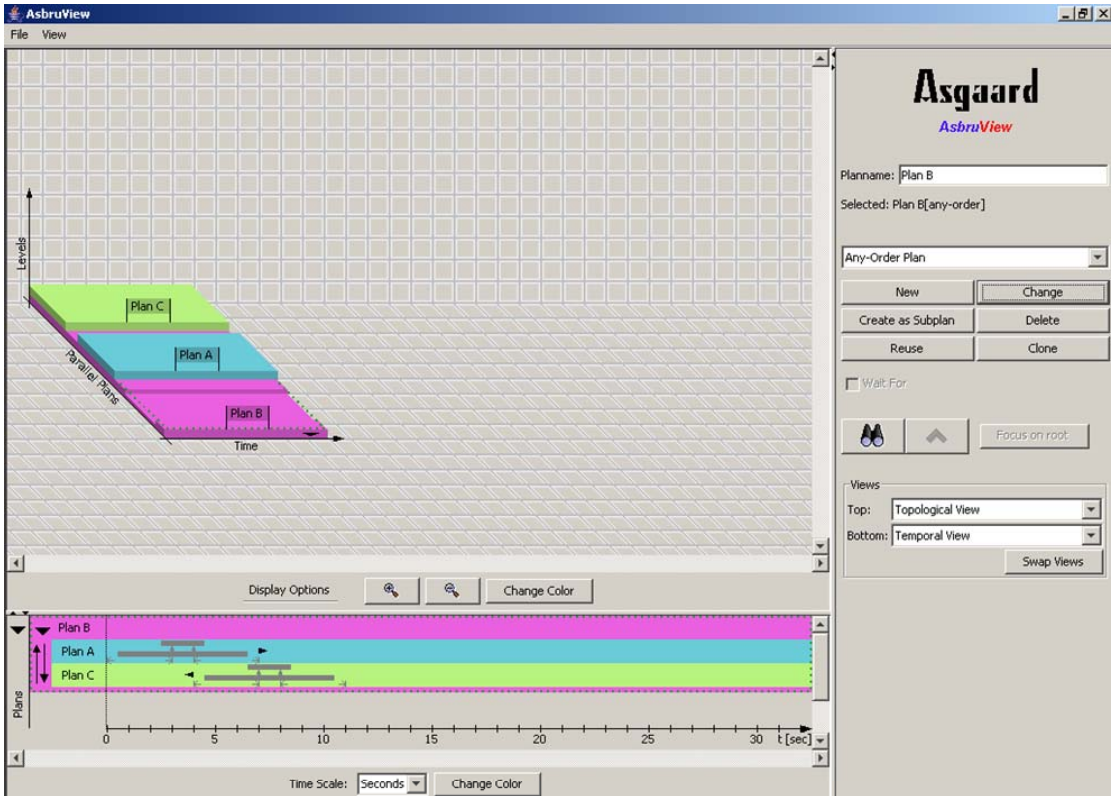

Figure 11 Specification of Critical Section in Asbru/AsbruView