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: A multitude of languages used for specification of Computer Interpretable Guidelines (CIG's) differ in their approaches to addressing particular modeling challenges. This paper addresses the lack of well defined semantics of such languages and absence of a single standard for their evaluation. We evaluated clinical guidelines modeling languages using the Workflow Patterns as an analysis framework. The analysis provided for clinical guidelines aims at an unbiased and vendor-independent survey of the expressive power of some modeling languages for representing the control-flow aspect of clinical guidelines.

Design: The pattern-based analysis has been applied to 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 the control-flow patterns.

Results: The results from the survey could be used for the selection of a language for representing a guideline, as well as for motivation and input to further development of any of the surveyed languages.

Conclusion: The evaluation shows that CIG modeling languages are remarkably close to traditional workflow languages. Moreover, as our paper shows, our analysis could help in the improvement and clarification of existing languages in this domain.

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. These types of systems have been shown to affect clinicians' behavior more than paper-based guidelines [1]. Although many parties have been engaged in developing *Computer-Interpretable Guidelines* (CIG's), there is still little standardization to facilitate sharing or to enable adaptation to local practice settings [2].

CIG's use “Task-Network Models” [3] for representation but differ in their approaches to addressing particular modeling challenges. In [2] authors compared six guideline modeling languages: Asbru, EON, GLIF, GUIDE, PRODIGY, and PRO*forma* according to eight components that capture the structure of CIG's. Although the authors sought for an objective way of comparing the models, the approach that they had applied for analysis of the guideline modeling languages was rather ad-hoc than systematic. In [4] authors took a life-cycle approach to compare five guideline formalisms: Asbru, EON, GLIF, PRO*forma*, and the Arden Syntax, examining guideline representation, acquisition, verification and execution aspects. *In this paper*, we examine the modeling languages using control-flow patterns. This is a big challenge because the semantics of these languages is incompletely and informally defined. Since CIG's focus on the ordering of tasks and activities, we focus on the control-flow perspective. Our analysis is done by means of revised control-flow patterns [5] that were previously defined by van der Aalst [6].

Workflow patterns have become a standard for assessing strengths and weaknesses of process specifications. The patterns have inspired the improvement and development of 10 languages and tools [20]. Furthermore, the workflow patterns were used for selecting a workflow management system and have been used in teaching [20].

We decided to analyze the current versions of the same set of languages as considered in [3]: 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 an active project. For each of the languages we identify whether workflow control-flow patterns can be supported by modeling facilities of the language. To illustrate how the supported patterns are implemented in practice we present screenshots of models designed respectively in Asbru/AsbruView, EON/*Protege-2000*, GLIF/*Protege-2000*, and PRO*forma*/Tallis.

The rest of the paper is organized as follows. The Background section gives an introduction to the computer-interpretable guidelines and describes the main features of the analyzed modeling languages. In the section Research Question we formulate the research questions we would like to answer in the given research context. The Methods section elaborates on the details of the performed evaluation, illustrating how selected patterns can be realized by the chosen set of CIG modeling languages. The Discussion section describes the differences and commonalities between the evaluated languages, and compares them using the control-flow patterns as the analysis framework. This paper ends with conclusions.

II. Background

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

A. Computer-Interpretable Guidelines

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 is consulted by 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 [7]. AsbruView is one among several tools (Delt/A [8,9], URUZ [10,11], and CareVis [12,13]) that were developed to support authoring of guidelines in Asbru [10]. A process model in Asbru [14] 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). AbsruView 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 [15] are scenarios, action steps, branching, decisions, and synchronization [16,17]. 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 [18] 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. A 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* [19] 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, preconditions, and postconditions. 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 the 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.

Table 1 illustrates terms used in the CIG modeling languages which correspond to the main workflow concepts that will be used throughout this paper.

B. Related Work

The recent *Workflow Patterns initiative* [20] 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 [21]. 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 were 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 [22,23]. 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 [24]. Furthermore, we revised the current set of the control-flow patterns and extended it with new patterns. We used the revised set of the control-flow patterns [5] to evaluate CIG's modeling languages.

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 [20]. The evaluation of the CIG’s modeling languages performed in this paper can contribute to standardizing of guideline-based decision support systems. For instance, a common agreement can be reached by selecting a set of mandatory and desirable patterns that have to be supported by modeling languages. CIG's are considered as a means for specifying guideline knowledge formally [33]. Formal representation of clinical guidelines helps to reduce ambiguities. Many formalisms exist for specifying CIG's, which are based on different often implicit assumptions, motivations and features [18,24,25,19,10,28,29,30,24].

The current evaluation can be considered inline with several comparisons performed by [3,31,32,34].Wang et al. [32] 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. In their comparison Tu and Musen [34] focused on the computational methods of the formalisms. De Clercq et al. addressed in their comparison guideline representation issues and some aspects of guideline acquisition, verification and execution [31]. Peleg et al. [3] analyzed CIG’s by examining them against identified by them eight dimensions capturing the conceptual components of CIG's. The CIG community aims at adopting a single CIG formalism as a standard for authoring, validation, execution and maintenance of CIG's. Not only does this require the involvement of different parties, but it also necessitates an achievement of an agreement between the parties upon the components inherent to CIG's. The results of the evaluation presented in this paper can be used for identifying the concepts and features that CIG's should support from the control-flow perspective.

III. Research Question

The main research questions which we want to answer in this research 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?". Furthermore, the results of the evaluation should help in verifying the following hypothesis: *"The main modeling entities offered by the clinical guideline modeling languages are very specific and not similar to the ones used for the process description in Workflow Management Systems".*

IV. Methods

In this section we define the criteria used for evaluating the pattern support and describe facilities offered by the examined CIG modeling languages to support a selection of patterns.

A. Evaluation criteria

For each model we checked whether it is possible to realize the control-flow pattern with the facilities offered by the model. 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. 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 support.

B. Analysis

The control-flow patterns that we used for evaluation can be logically divided into several groups: the basic control-flow patterns, the advanced branching and synchronization patterns, the structural patterns, the multiple instances patterns, the state-based patterns, the cancellation patterns and a set of 23 new patterns identified as a result of the revision of the classical set of control-flow patterns. We examined the language specification and checked how the selected tools support the patterns. We illustrate the pattern support using a selection of patterns that received different ratings for all examined offerings; patterns that were either supported or unsupported by all four formalisms are described online at http://www.bpmcenter.org/reports). Table 2 summarizes the support of the full set of 43 patterns by the four CIG modeling-languages. The patterns described in detail in this paper refer to the numbering used in Table 2.

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

Advanced Branching and Synchronization Patterns correspond to advanced branching and synchronization scenarios: *Multiple Choice, Synchronizing Merge*, *Multiple Merge*, and *Discriminator*. While basic control-flow patterns enable to do 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.

Pattern *Structured Synchronizing Merge c*ombines 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.

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. 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.

Structural patterns: *Arbitrary cycles* and *Implicit termination* identify whether the modeling formalism has any restrictions regarding the structure of the processes.

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). Similar structure can be built in EON. Asbru supports only structured loops 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).

The Multiple Instances 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 other patterns related to the notion of Multiple Instance task (i.e., we exclude patterns 26 and 27 Cancel and Complete Multiple Instance Activity respectively).

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*.

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. Note that although in EON a *Choice Step* and the associated *Action Choice* presents 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. 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* block 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*.

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).

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 one of the following types: end-of-previous step, patient-data-availability, patient-arrival, or temporal. Similar to EON, GLIF has *Patient State* step which denotes the state of the patient.

Patient State step is used to denote multiple entry points to a guideline model. PRO*forma* supports this pattern by means of a state trigger that allows checking states of activities and values of truth-valued expressions.

The Cancellation patterns contain two patterns: Cancel Activity and Cancel Case. Cancel Activity pattern is supported by all examined offerings. Exceptional is the support of cancelling an arbitrary set of tasks provided by Asbru.

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. EON does not support this pattern. 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. PRO*forma* supports this pattern via an abort condition associated with a plan containing all activities.

The new 23 control-flow patterns consist of a set of new patterns and a number of the revised pattern variants described earlier. Among them are patterns which address the concepts of a trigger, path and thread branching and synchronization, cancellation.

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.

Asbru and EON do not support this pattern. 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.

Pattern 29. Cancelling 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 also qualifies as a support of the special case 1-out-of-M Join known as the *Discriminator* pattern.

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.

Although EON supports 1-out-of-M join, it does not support this pattern. Asbru supports the pattern by *wait-for-group* attribute of a plan that specifies that *n* task 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).

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

Description The convergence of two or more branches into a single subsequent branch following one or more corresponding divergencies 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 specifications 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 specifications 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.

EON and PRO*forma* do not support this pattern. 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.

V. Results

In this section, we provide a detailed examination of the modeling languages. Table 2 summarizes the comparison.

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. The lowest degree of pattern support has been shown by EON (it supports only 11 patterns).

More detailed analysis of the pattern support revealed that all examined offerings directly support Basic Control-flow patterns. At least half of the Advanced Branching and Synchronization patterns that 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. In the business processes domain, multiple instances 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). 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.

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*. Very good support of Cancellation patterns has been provided by all analyzed languages.

So far, workflow control-flow patterns have been used to evaluate Workflow Management Systems and business standards. Process models in Workflow Management Systems can be designed for different types of business processes and thus can be applied in various domains. The main research question that we wanted to answer in this research is: "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?". *Our initial hypothesis that main modeling entities offered by the clinical guideline modeling languages are very specific and not similar to the ones used for the process description in Workflow Management Systems has been rejected by our findings.* The results of the evaluation of the clinical guidelines modeling languages showed that from the control-flow perspective these languages are very similar to the process languages of Workflow Management Systems (WFMS). This is remarkable since one would have expected dedicated constraints allowing for more flexibility.

VI. Discussion

Members of the medical community have always emphasized that they want to use their own guideline formalisms rather than base the control-flow model on existing workflow formalisms from the organizational domains, because guidelines should be flexible. However, when we examined guideline modeling languages we did not find more flexibility than in industrial cases. Based on this, the medical community might rethink about adopting known formalisms for expressing control flow of guideline models. For instance, the case-handling system FLOWER offers a high degree of flexibility during the execution of a case (i.e., a process instance). Although it 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.

On the other hand, guideline modeling languages offer strong support for expression languages that support modeling of complex decisions. They also provide ways for modeling decision 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 in 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, this support of multiple entry point is not unique and has alternatives. Similar behavior can be achieved by means of the state triggers in PRO*forma*.

When it comes to the notion of flexibility, the medical community considers as important flexibility for specifying the process as well as flexibility provided during the process execution. This means that techniques must be developed which will allow the integration of guideline-based recommendations into the management of health-care processes in ways that are sensitive to the needs of specific patients and specific health care organizations [34]. To increase the degree of flexibility, the medical community might consider extending the guideline modeling-languages with constructs from workflow formalisms that define the semantics of flexible constructs. This may lead eventually to defining a new control-flow component for CIG languages that could become a standard. In addition to the set of constructs discussed in this paper, the medical community may consider using also configurable modeling constructs, found in business process formalisms [35]. Configurable modeling constructs could enable specifying ahead of time how a CIG that was encoded by the institution that developed could potentially be changed by an institution that wants to implement the CIG, adjusting it to its local organization. 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 [36]. 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 CIG's process model.

The results of the evaluation 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 means of the CIG's may match the list of their requirements against the results of the pattern-based evaluation. If an organization requires exclusive execution of activities in non-predefined order, then Asbru might be chosen, since no other languages 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).

While evaluating the modeling languages, we have identified several scenarios which are not covered by the set of the control-flow patterns we 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). 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 on the moment of triggering. We will augment the current set of patterns with the patterns covering the above-described scenarios.

VII. Conclusions

In this paper, we have thoroughly examined the CIG's modeling languages Asbru, EON, GLIF3.5 and PRO*forma* to identify the degree of support of the control-flow patterns. We identified differences and similarities of the languages between each other and determined their similarity to the process modeling languages of Workflow Management Systems. We also showed how on the basis of a given problem a single language among considered ones can be selected. Furthermore, we indicated the possible directions for achieving more flexibility by the guideline modeling languages.

VIII. 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.

References

1. S. Shea, W. DuMouchel, and L. Bahamonde. A Meta-analysis of 16 Randomized Controlled Trials to Evaluate Computer-based Clinical Reminder Systems for Preventive Care in the Ambulatory Setting. J Am Med Info, 3(6):399–409, 1996.

2. M. Peleg. Chapter 4-2: Guideline and Workflow Models. In Greenes R.A., editor, Medical Decision Support: Computer-Based Approaches to Improving Healthcare Quality and Safety. Elsevier, 2006.

3. M. Peleg, S.W. Tu, J. Bury, P. Ciccarese, J. Fox, and et al. R.A. Greenes. Comparing computer interpretable guideline models: A case-study approach. page 10(1):5268. J Am Med Inform Assoc., 2003.

4. 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, (31), 2004.

5. N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View (Draft version; request for comments). BPM Center Report BPM-06-22, BPMcenter.org, 2006.

6. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(1):5–51, 2003.

7. AsbruView. http://www.ifs.tuwien.ac.at/asgaard/asbru/tools.html.

8. DELT/A. Document Exploration and Linking Tool/Addons. http://ieg.ifs.tuwien.ac.at/projects/delta.

9. P. Votruba, S. Miksch, and R. Kosara. Facilitating Knowledge Maintenance of Clinical Guidelines and Protocols. In 11th World Congress of Medical Informatics, 2004.

10. 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.

11. R. Kosara and S. Miksch. Metaphors of Movement: A Visualization and User Interface for Time-Oriented Skeletal plans. Artif Intell Med, (22), 2001.

12. CareVis. http://ieg.ifs.tuwien.ac.at/projects/carevis.

13. W. Aigner and S. Miksch. CareVis: Integrated Visualization of Computerized Protocols and Temporal Patient Data. Artif Intell Med, (37), 2006.

14. 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.

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

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

17. S.W. Tu. The eon guideline model. Technical report, http://smi.stanford.edu/projects/eon/EONGuidelineModelDocumentation.doc, 2006.

18. 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.

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

20. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns Home Page. http://www.workflowpatterns.com.

21. 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.

22. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

23. 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.

24. 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, Center Report BPM-06-18, BPMcenter.org, 2006.

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

26. S. Tu, J. Campbell, and M.A. Musen. The SAGE guideline modeling: motivation and methodology. Stud Helath Technol Inform., 101:162–166, 2004.

27. P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, G. Molino, and G. Correndo. The GLARE approach to clinical guidelines: main features. Stud Health Technol Inform., 101:162–166, 2004.

28. 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., pages 389–393, 2000.

29. 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.

30. P. Ciccarese, E. Caffi, S. Quaglini, and M. Stefanelli. An innovative Health Information System: The Guide Project. Int J Med Info, 2004.

31. P.A. de Clercq, J.A. Blom, H.H. Korsten, and A. Hasman. Approaches for creating computer interpretable guidelines that facilitate decision support. Artific Intel med, 31:1–27, 2001.

32. 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.

33. 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, pages 125–142, Leipzig, Germany, 2000.

34. A Kumar, B Smith, D Pisanelli, A Gangemi, M. An Ontological Framework for the Implementation of Clinical- Studies In Health Technology And Informatics, 2004

35. Wil M. P. van der Aalst, Alexander Dreiling, F. Gottschalk, Michael Rosemann, M. H. Jansen-Vullers: Configurable Process Models as a Basis for Reference Modeling. Business Process Management Workshops 2005: 512-518

36. 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.

Terms	**Asbru**	**EON**	**GLIF**	**PRO*forma***
Process model	Plan	Guideline	Guideline	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 1 Terms used by 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. Cancelling Discriminator	+	-	-	+
30. Structured N-out-of-M Join	+	-	+	+
31. Blocking N-out-of-M Join	-	-	-	-
32. Cancelling 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	-	-	-	-

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

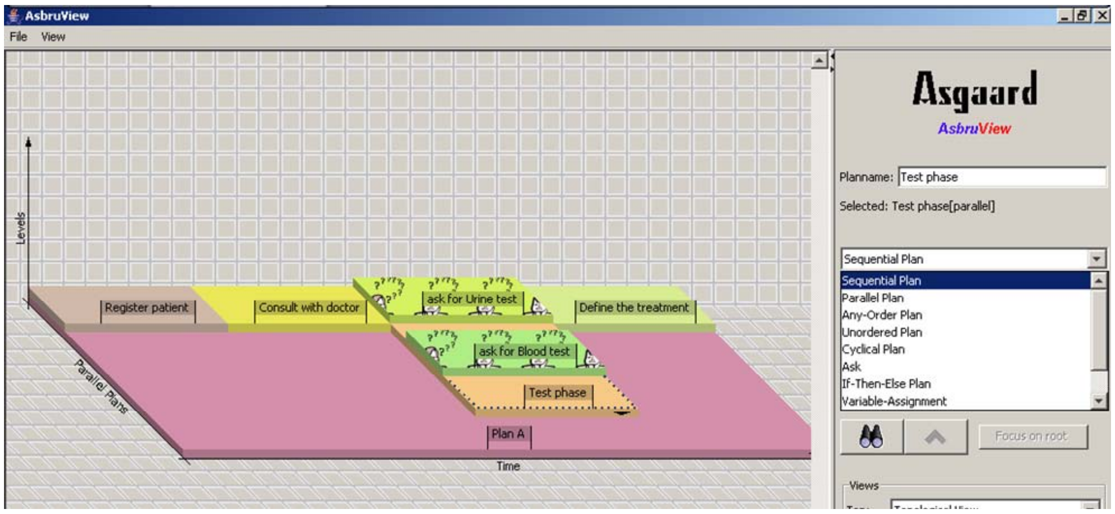


Figure 1 The patient-diagnosis scenario modeled in AsbruView

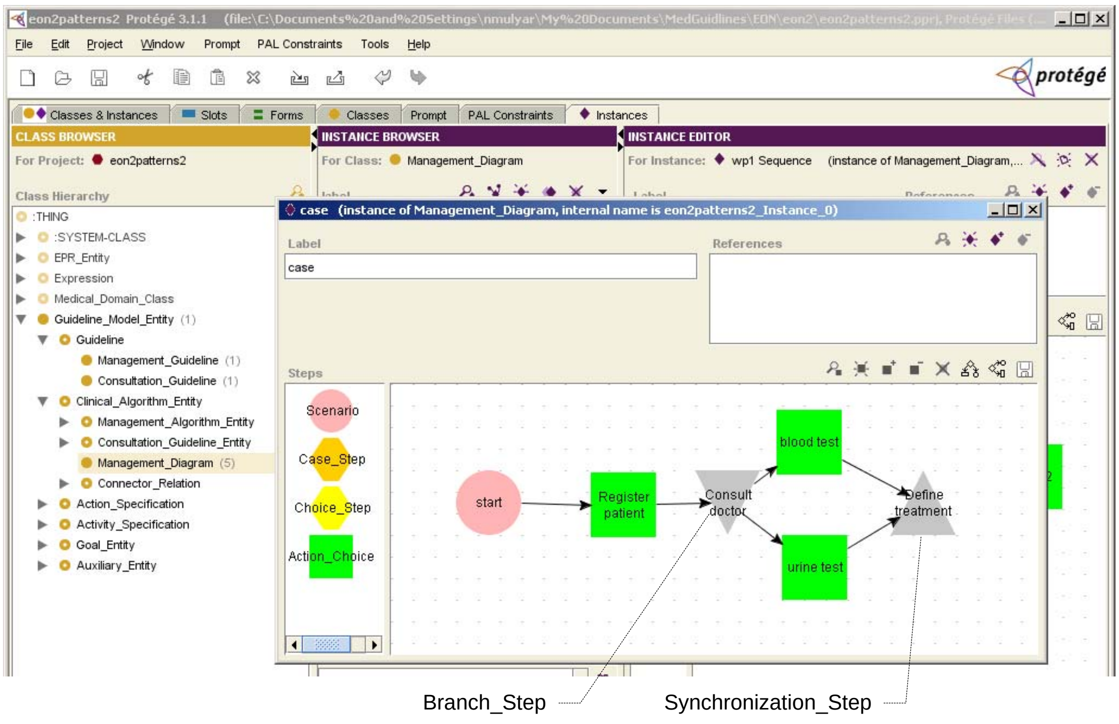

eon2patterns2 Protégé 3.1.1 (file:\C:\Documents%20and%20Settings\nmulyar\My%20Documents\MedGuidlines\EON\eon2\eon2patterns2.pprj, Protégé Files (...
File Edit Project Window Prompt PAL Constraints Tools Help
protégé
Classes & Instances
Slots
Forms
Classes
Prompt
PAL Constraints
Instances
CLASS BROWSER
For Project: eon2patterns2
Class Hierarchy
:THING
:SYSTEM-CLASS
EPR_Entity
Expression
Medical_Domain_Class
Guideline_Model_Entity (1)
Guideline
Management_Guideline (1)
Consultation_Guideline (1)
Clinical_Algorithm_Entity
Management_Algorithm_Entity
Consultation_Guideline_Entity
Management_Diagram (5)
Connector_Relation
Action_Specification
Activity_Specification
Goal_Entity
Auxiliary_Entity
INSTANCE BROWSER
For Class: Management_Diagram
INSTANCE EDITOR
For Instance: wp1 Sequence (instance of Management_Diagram,...
case (instance of Management_Diagram, internal name is eon2patterns2_Instance_0)
Label
case
References
Steps
Scenario
Case_Step
Choice_Step
Action_Choice
start
Register patient
Consult doctor
blood test
urine test
Define treatment
Branch_Step
Synchronization_Step

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

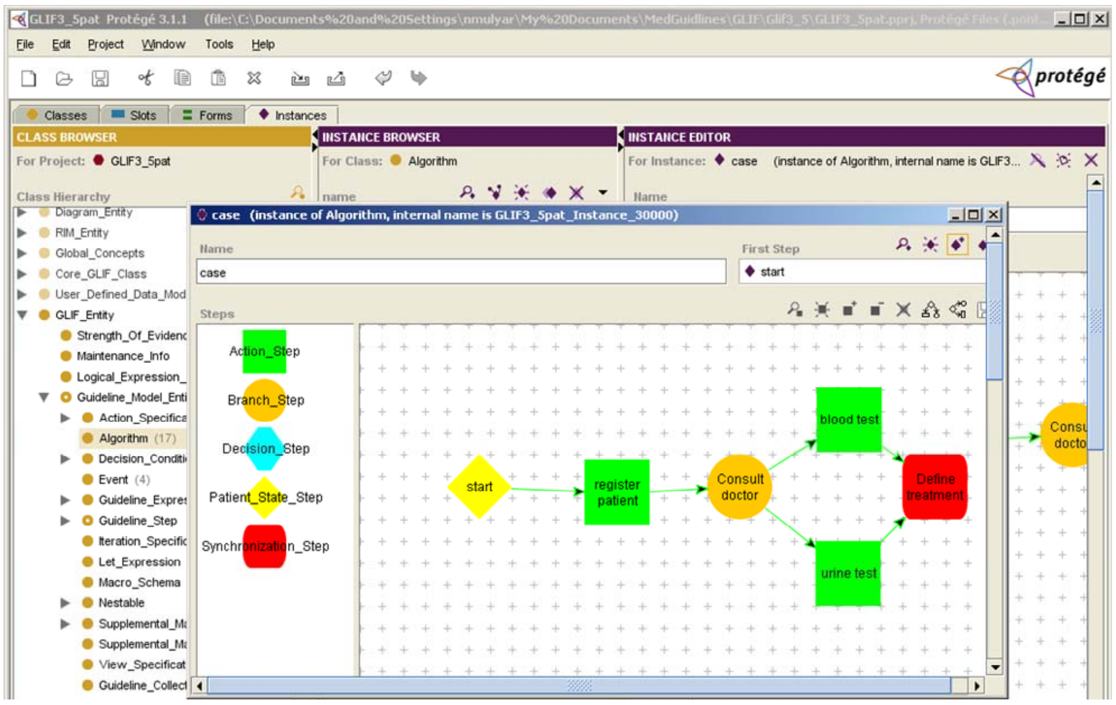


Figure 3 The patient-diagnosis scenario modeled in GLIF3.5/Protege

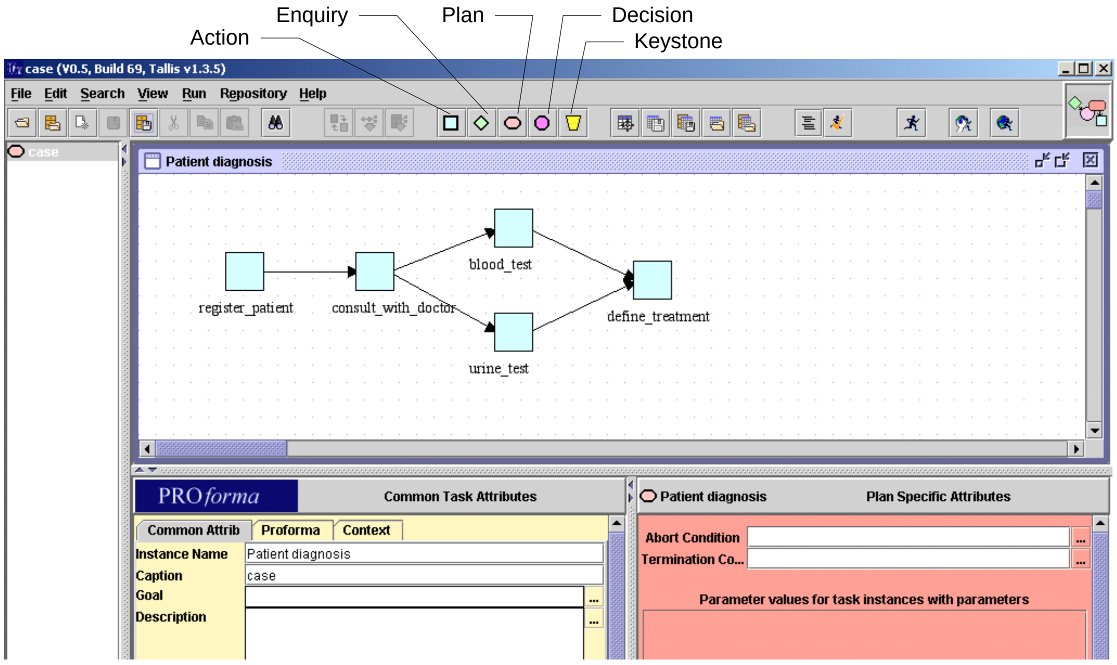


Figure 4 The patient-diagnosis scenario modeled in PRO*forma*/Tallis

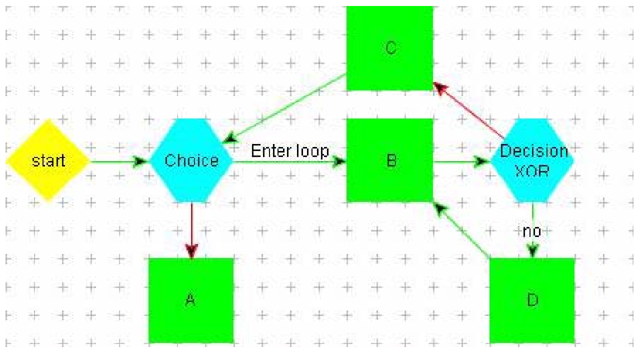


Figure 5 Specification of the Arbitrary Cycles in GLIF3.5/Protege

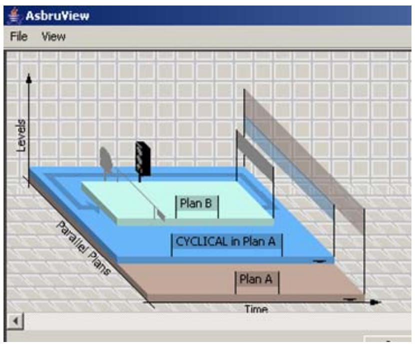
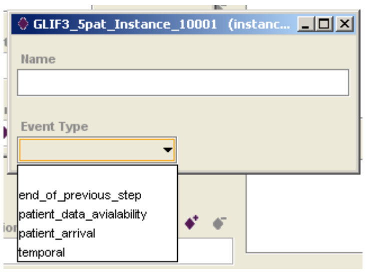
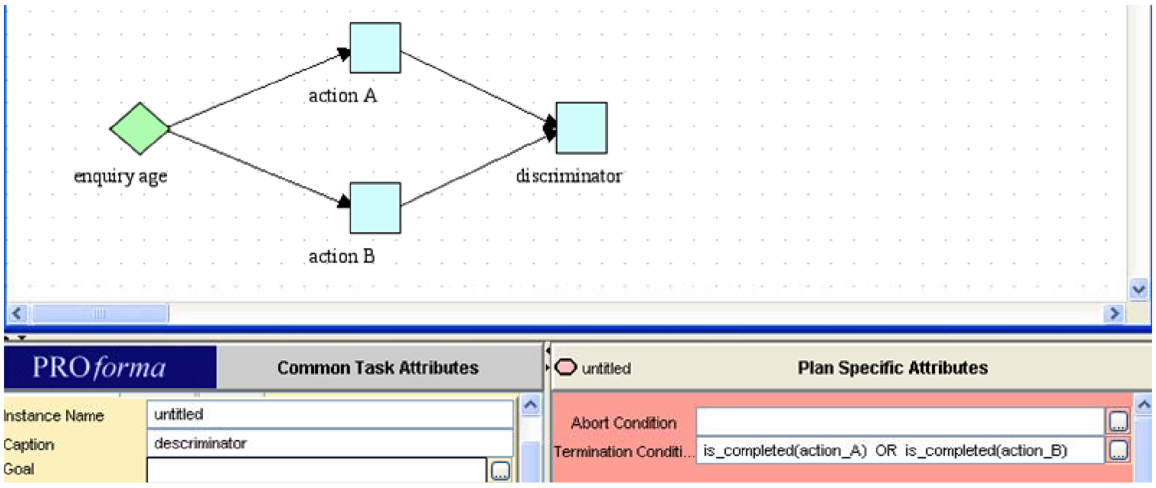


Figure 6 Specification of the Structured Loop in Asbru/AsbruView



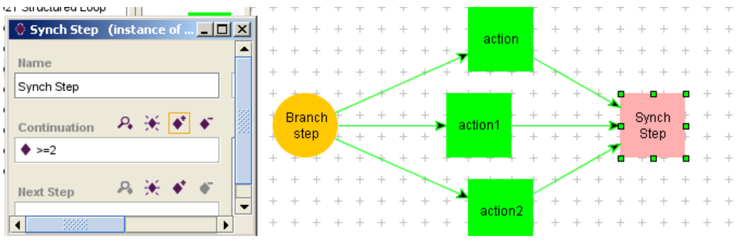
end_of_previous_step
patient_data_avialability
patient_arrival
temporal

Figure 7 Specification of the Persistent Trigger in GLIF3.5



Instance Name	untitled
Caption	descriminator
Goal	
Abort Condition	
Termination Conditi...	is_completed(action_A) OR is_completed(action_B)

Figure 8 Specification of the Cancelling Discriminator in PROforma/Tallis

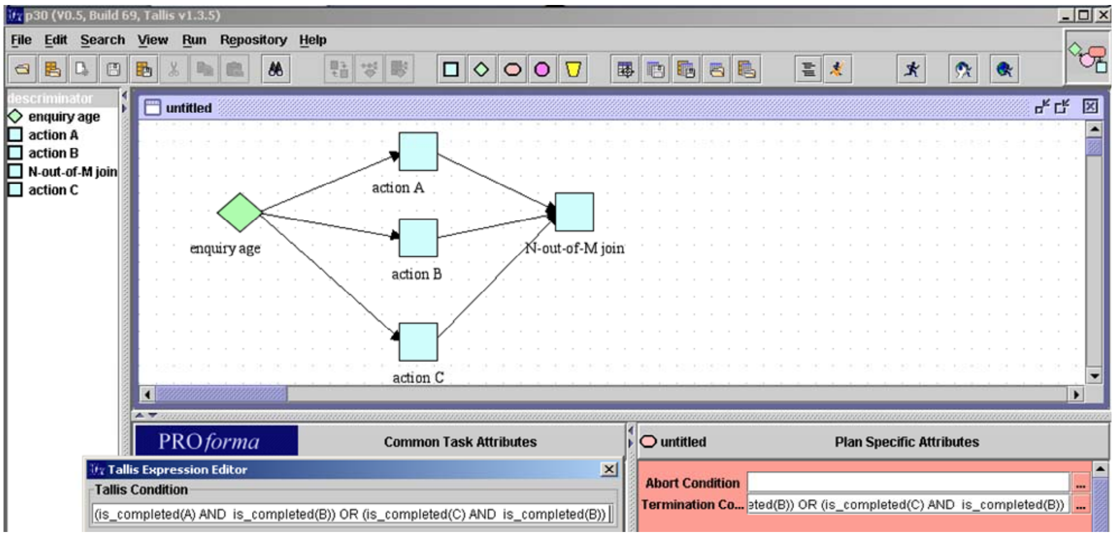


Name: Synch Step

Continuation: >=2

Next Step:

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

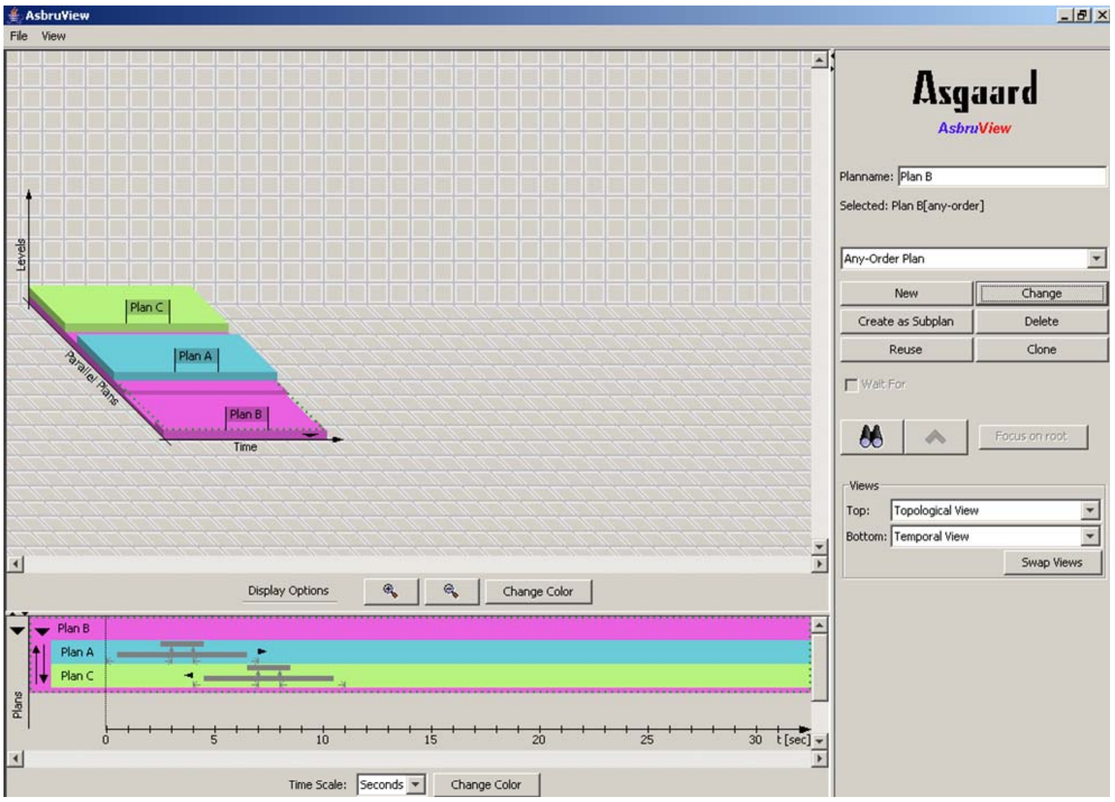


Tallis Condition: (is_completed(A) AND is_completed(B)) OR (is_completed(C) AND is_completed(B))

Abort Condition:

Termination Co...: ated(B)) OR (is_completed(C) AND is_completed(B))

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



Planname: Plan B

Selected: Plan B[any-order]

Any-Order Plan

Top: Topological View

Bottom: Temporal View

Time Scale: Seconds

Figure 11 Specification of Critical Section in Asbru/AsbruView