Asbru's Execution Engine: Utilizing Guidelines for Artificial Ventilation of Newborn Infants

Christian Fuchsberger and **Silvia Miksch**
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188, A-1040 Vienna, Austria
email: {christian, silvia}@asgaard.tuwien.ac.at

Abstract

In real-world environments of neonatal intensive care units (NICUs), a major challenge in using clinical guidelines and protocols in a computer readable form is to provide (i) a clear, precise representation to handle the complex forms of uncertainty and time-critical situations which are common in the ICU domain and (ii) an execution engine to be able to tackle such a representation.

This paper presents an execution unit tailored to a particular guideline-representation language called Asbru. Asbru is able to capture the required features to proceed with guideline-based care within NICUs. We have evaluated our execution engine by modelling the artificial ventilation of newborn infants in Asbru and by applying our execution engine to real patient data and Asbru plans.

1 Introduction

In the last couple of years a huge amount of clinical guidelines and protocols were published to improve and to standardize medical treatment planning. However, the clinical treatment plans in Neonatal Intensive Care Units (NICUs) available are limited. Due to the information overload resulting from the modern technical equipment the medical staff can hardly deduce optimal procedures for concrete treatment. The electronic data processing could help to access the knowledge pool, but in this domain where only completely reliable results count and more over every mistake can lead to catastrophic results, the scepticism regarding computer aided systems is very high.

To overcome such an attitude of doubt, computer-supported systems were developed which assure that the human (the medical staff) is executing the actions ("open-loop" methodology) instead of automatic control of the devices ("closed-loop" methodology). Nevertheless, the critical part is the modelling of the medical knowledge (in the form of clinical guidelines and protocols) and the execution of such medical knowledge.

To structure and ease the modelling part, various guideline and protocol-representations languages were introduced [11]. One representative is the Asbru language [9; 13], which is a time-oriented, intention-based, skeletal plan-representation language that is used in the *Asgaard* Project [8; 14] to represent clinical guidelines and protocols.

To support the execution of clinical guidelines and protocols an execution engine is needed according to the complexity or power of the guideline-representation language. In the last couple of years a few execution engines were introduced [11; 15]. However, to capture the complexity of clinical guidelines a powerful engine is needed. Our principle assumption is that a very powerful plan-representation language as Asbru is needed to represent clinical guidelines. We have developed an execution engine, which is able to handle a specific class of guidelines written in Asbru. This class of Asbru plans are oriented to the specific needs and demands in NICUs. The execution is monitored and controlled by a set of modules called Asbru Run-Time Modules (AsbruRTM) which give advice to medical staff based on the procedures laid down in such Asbru plans [9; 13].

In the first section, we will illustrate related work in the problem area. The second section will clarify which kind of plan representation we have used, namely Asbru. The core of the paper is the following three sections, first, the architecture of the execution engine, second the concrete implementation of modules involved and third, how these modules are utilized for the controlled ventilation of newborns.

2 Related Work

Guideline- and Protocol-based care was improved due to the computer-support knowledge acquisition, modelling and execution. In the last couple of years various formal and semi-formal representations of medical guidelines and protocols were introduced to ease the structuring and modelling parts [11]. These representations range from workflow representation approaches like Guide [12], over scenario-based ones, like PRODIGY [7] to sophisticated and complex representations like GLIF [10] or ProForma [3].

The major challenge in representing clinical protocols in a computer readable form is to both provide a clear, precise representation with defined semantics and to handle the complex forms of uncertainty and time-critical situations which are common in the ICU domain. Often these representations provide a clearly defined framework but the frames are filled with free text. Such a protocol can therefore, only be interpreted by a human and not by a computer. Also execution or verification can only be performed by humans who have to interpret each part of free text and decide its precise meaning - an unreliable and often not reproducible process. There are numerous notations of logic which provide clear formal semantics. However, the task of modelling a protocol in such an annotation is simply impossible to achieve. In particular, intertwined processes which develop over time and which involve uncertainty are hard to model in formal logic from scratch. In contrast, the plan-representation language Asbru (compare Section 3) developed within the Asgaard project has clear defined semantics and complex language constructs to represent uncertain and incomplete knowledge. It is therefore, suitable to capture a protocol in a formally defined way while maintaining maximum readability and user/developer friendliness.

An execution engine, which is able to handle the complexity of Asbru plans, is very complicated due to the powerful features of Asbru. An interpreter for Asbru was developed by Tibor Bosse [2]. It consists of two modules. One of them is a parser written in prolog which maps the XML Asbru plan to clips facts. The other is the main program using the generated facts to simulate the execution of the Asbru plan. The interpreter supports only a subset of Asbru (called Asbru Light). A further disadvantage of the interpreter is the missing of real time handling and adequate implementation of time constrains.

In the next section we will clarify which features of Asbru are tackled by the proposed execution engine called AsbruRTM (Asbru Rund-Time-Mudule).

3 Plan-Representation Language

We will give a very short introduction to Asbru and illustrate the subset, which is covered in Asbru Light+.

3.1 A Short Introduction to Asbru

Asbru [9; 13] is a time-oriented, intention-based, skeletal plan-representation language that is used in the *Asgaard* Project[1] to represent clinical guidelines and protocols. Asbru can be used to express clinical protocols as skeletal plans [5] that can be instantiated for every patient (for an example see Fig. 8). It was designed specific to the set of plan-management tasks [8]. Asbru enables the designer to represent both the prescribed actions of a skeletal plan and the knowledge roles required by the various problem-solving methods performing the intertwined supporting subtasks. The major features of Asbru are that

- prescribed actions and states can be continuous;
- intentions, conditions, and world states are temporal patterns
- uncertainty in both temporal scopes and parameters can be flexibly expressed by bounding intervals;
- plans might be executed in sequence, all plans or some plans in parallel, all plans or some plans in a particular order or unordered, or periodically;
- particular conditions are defined to monitor the plans' execution; and
- explicit intentions and preferences can be stated for each plan separately.

In Asbru, the following parts of a plan can be specified: *preferences*, *intentions*, *conditions*, *effects*, and *plan body (actions)*.

Preferences. Preferences constrain the applicability of a plan (e.g., select-criteria: exact-fit, roughly-fit) and express the kind of behaviour of the plan (e.g., kind of strategy: aggressive or normal).

Intentions. Intentions are high-level goals that should be achieved by a plan, or maintained or avoided during its execution. This information is very important not only for selecting the right plan, but also for critiquing treatment plans as part of the ever ongoing process of improving the treatment. This makes intentions one of the key parts of Asbru.

Conditions. Conditions need to hold in order for a plan to be *started*, *suspended*, *reactivated*, *aborted*, or *completed*. Two different kinds of conditions (called preconditions) exist, that must be true in order for a plan to be started: filter-preconditions cannot be achieved through treatment (e.g., subject is female), setup-preconditions can. After a plan has been started, it can be suspended (interrupted) until either the restart-condition is true (whereupon it is continued at the point where it was suspended) or it has to be aborted. If a plan is aborted, it has failed to reach its goals. If a plan completes, it has reached its goals, and the next plan in the sequence is to be executed.

Effects. Effects describe the relationship between plan arguments and measurable parameters by means of mathematical functions. A probability of occurrence is also given.

Plan Body (Actions). The plan body contains plans or actions that are to be performed if the preconditions hold. A plan is composed of other plans, which must be performed according to the plan's type:

[1]In Norse mythology, *Asgaard* was the home of the gods. It was located in the heavens and was accessible only over the rainbow bridge, called *Asbru* (or *Bifrost*) (For more information about the *Asgaard* project see `http://www.asgaard.tuwien.ac.at`).

in *sequence*, in *any order*, in *parallel*, *unordered*, or *periodically* (as long as a condition holds, a maximum number of times, and with a minimum interval between retries).

A plan is decomposed into sub-plans until a non-decomposable plan — called an action or a user-performed plan — is found. All the sub-plans consist of the same components as the plan, namely: preferences, intentions, conditions, effects, and the plan body itself.

Plans are executed (i.e., their parameters monitored, conditions checked and reacted to) by an execution unit. User-performed plans are displayed to the user so that he or she can react and then tell the machine if and when the action is finished and if it was successful.

Time Annotations. An important part in specifying the complex temporal aspects of plans are Time Annotations. A Time Annotation specifies four points in time relative to a reference point (which can be a specific or abstract point in time, or a state transition of a plan): The earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS) and latest finishing shift (LFS). Two durations can also be defined: The minimum duration (MinDu) and maximum duration (MaxDu). Together, these data specify the temporal constraints within which an action must take place, or a condition must be fulfilled for a condition to trigger (compare Figure 1).

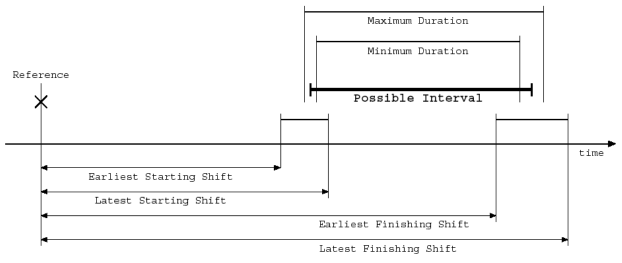


Figure 1: Asbru's Time Annotations

3.2 Asbru Light+

The starting point for the selection of Asbru Light+ was the definition of Asbru Light by Tibor Bosse [2], which was used in the ProtoCure project[2]. However, the usage of this subset showed that this subset has a lot of limitation to be applicable to the dynamically changing and time-critical domain of intensive care units (compare Section 2). Therefore, we extended Asbru Light to Asbru Light+ which embodies all necessary parts to handle controlled ventilation of newborns. Asbru Light+ covers the handling of temporal uncertainty and real time constraints as well as the following Asbru parts: intentions, conditions, and the plan body including the various plan layouts.

[2]For more information about the *ProtoCure* project see `http://www.protocure.org`.

4 Architecture of the Asbru Execution Engine

In this section we present the three core modules for Asbru execution engine: data abstraction, environment monitoring and execution.

4.1 Data-Abstraction Unit

The Data-Abstraction Unit is the connecting part between the hospital information system, the sensor data, the user input and the Monitoring Unit. Further the unit carries out three useful tasks: it validates the obtained data, it calculates derived values and it transforms quantitative values into qualitative ones.

4.2 Environment Monitoring Unit

The Environment Monitoring Unit bridges the gap between the Data-Abstraction and the Execution Units. It gets a list of available parameter propositions form the Data-Abstraction Unit and adds them to a list of observed parameter propositions (OPP's). During the plan execution the execution unit specifies a list of temporal patterns which are key to future state transitions of instantiated plans. These patterns are denoted monitored parameter propositions (MPP). Whenever an OPP matches a MPP, the execution unit is notified.

4.3 Execution Unit

The Execution Unit handles the state transitions of the plans. It instantiates plans from the plan library and governs their life-cycle according to the state of the world as reported by the monitoring unit.

The data abstraction is specified in the domain-section of the plans. Each of the abstraction steps is defined in one parameter definition. Thus, the data abstraction and the plan execution are both configured in a single file.

5 AsbruRTM: The Asbru-Run-Time Module

The AsbruRTM (Asbru-Run-Time Module) is an execution engine which is able to execute clinical protocols and guidelines written in Asbru plans. It consists of three core modules: the data abstraction, environment monitoring and execution unit (see Figure 2). Additionally, on the one hand, AsbruRTM has access to the various clinical protocols and guidelines stored in the plan library, on the other hand, a straightforward user interface was implemented for testing purposes. In the following section we describe the principles of AsbruRTM.

The AsbruRTM is written in Java according to the need of high platform independence, object-oriented design, and freely available XML components for parsing and mapping. Extern user interfaces written in

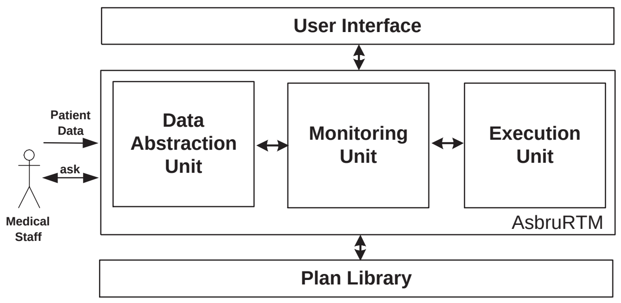


Figure 2: AsbruRTM's Framework

other programming languages than Java can though be served by a CORBA connection object.

5.1 Asbru Classes

Asbru is available in XML format [13] and we needed a straightforward transformation of these XML representation in Java classes. Therefore, we used the tool Castor [4] to generate the classes (approximately 3000 classes). Castor is an open source data binding framework for Java. It generates a Java object model out of an XML Schema.

5.2 Parallel Processing

AsbruRTM's primary requirement is parallel processing as all three core modules must simultaneously fulfil their tasks. Hence the data-abstraction unit has to evaluate continuously the values received by the sensors. In progression it informs the monitoring unit which continuously checks, whether an OPP maps a MPP. While meantimes the execution needs to go on. Figure 3 shows the parallel processing of a plan fragment from Table 1.

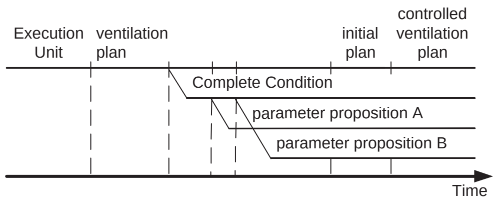


Figure 3: AsbruRTM's Parallel Processing. Horizontal lines stand for threads. Thread labels indicate their actual task. Branches represent the start of a new thread.

5.3 Execution logic

Asbru plans contain all needed information regarding the initialisation and execution of medical guidelines. Therefore, the main logic is provided by the Asbru classes generated by Castor. TABLE 1 displays this fact.

Plan Fragment

```
1 [...]
2 <plans>
3  <plan-group>
4   <plan name="ventilation_plan">
5    <conditions>
6     <complete-condition>
7      <constraint-combination type="and">
8       <parameter-proposition parameter-name="FiO2">
9        <value-description type="less-or-equal">
10         <numerical-constant value="40"/>
11         [...]
12    <plan-body>
13     <subplans type="sequentially">
14     [...]
15     <plan-activation>
16      <plan-schema name="initial_plan"/>
17     [...]
```

Actions of AsbruRTM

Line 4: The primary plan (here the "'ventilation_plan"') is started by the execution unit

Line 6: The "'Complete Conditon"' is started in its own thread

Line 7-8: It is created a thread-group for the "'parameter propositions"' threads.

Line 8-10: The thread of this "'parameter proposition"' is finished when FiO_2 is less 40 or the application is aborted.

Line 13: The following plans are started sequential, hence there are not created additionally threads for their executing.

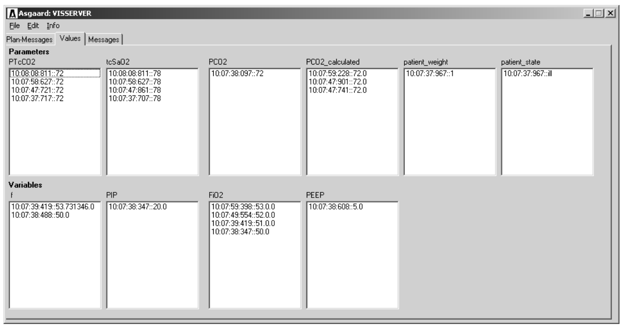
Line 16: The "'initial_plan"' is started.

Table 1: Execution of a Plan Fragment

5.4 Running a Test Case

In this section we explain the use of AsbruRTM. AsbruRTM is started in three steps:

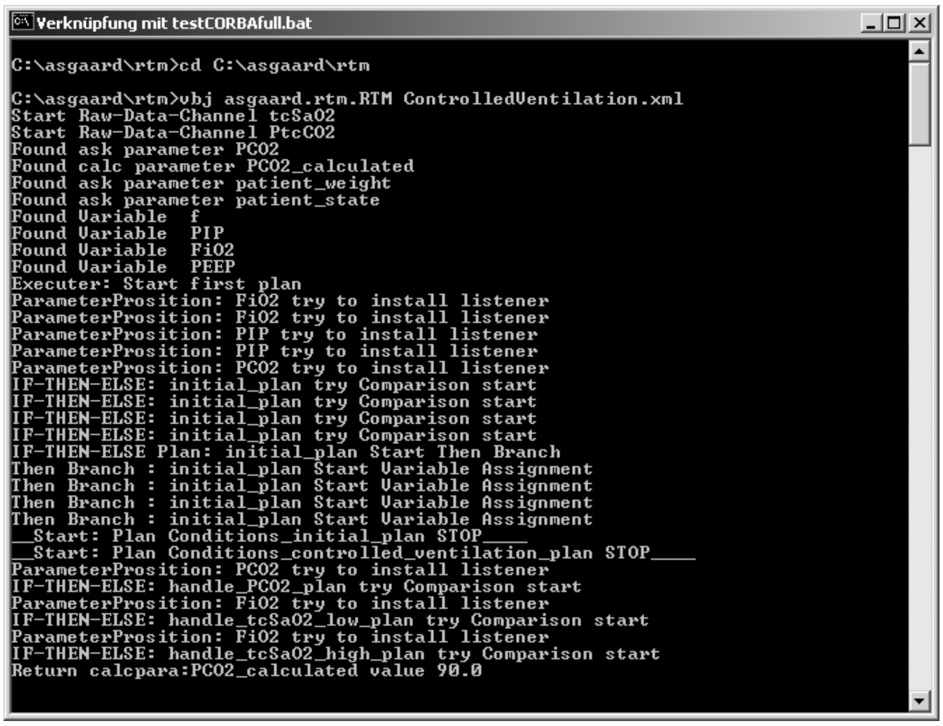
1. **CORBA:** The name service for CORBA is started.
2. **User interface:** The example user interface, programmed in Borland Delphi [1], is started (see Figure 4.



```
PTcCO2
10:08:08:811::72
10:07:58:627::72
10:07:47:721::72
10:07:37:717::72

tcSaO2
10:08:08:811::78
10:07:58:627::78
10:07:47:861::78
10:07:37:707::78

PCO2
10:07:38:097::72

PCO2_calculated
10:07:59:228::72.0
10:07:47:901::72.0
10:07:47:741::72.0

patient_weight
10:07:37:967::1

patient_state
10:07:37:967::ill

f
10:07:39:419::53.731346.0
10:07:38:488::50.0

PIP
10:07:38:347::20.0

FiO2
10:07:59:398::53.0.0
10:07:49:554::52.0.0
10:07:39:419::51.0.0
10:07:38:347::50.0

PEEP
10:07:38:608::5.0
```

Figure 4: AsbruRTM's User Interface

3. **AsbruRTM:** AsbruRTM is started with the desired plan as command line parameter. Then the plan is parsed and validated. Accordingly the corresponding Java objects are created from the XML plan. If this task is successful done the domain is initialised and the proper plan execution starts. Figure 5 shows the status-output of AsbruRTM.



```
Verknüpfung mit testCORBAfull.bat

C:\asgaard\rtm>cd C:\asgaard\rtm

C:\asgaard\rtm>vbj asgaard.rtm.RTM ControlledVentilation.xml
Start Raw-Data-Channel tcSaO2
Start Raw-Data-Channel PtcCO2
Found ask parameter PCO2
Found calc parameter PCO2_calculated
Found ask parameter patient_weight
Found ask parameter patient_state
Found Variable  f
Found Variable  PIP
Found Variable  FiO2
Found Variable  PEEP
Executer: Start first plan
ParameterProsition: FiO2 try to install listener
ParameterProsition: FiO2 try to install listener
ParameterProsition: PIP try to install listener
ParameterProsition: PIP try to install listener
ParameterProsition: PCO2 try to install listener
IF-THEN-ELSE: initial_plan try Comparison start
IF-THEN-ELSE: initial_plan try Comparison start
IF-THEN-ELSE: initial_plan try Comparison start
IF-THEN-ELSE: initial_plan try Comparison start
IF-THEN-ELSE Plan: initial_plan Start Then Branch
Then Branch : initial_plan Start Variable Assignment
Then Branch : initial_plan Start Variable Assignment
Then Branch : initial_plan Start Variable Assignment
Then Branch : initial_plan Start Variable Assignment
__Start: Plan Conditions_initial_plan STOP____
__Start: Plan Conditions_controlled_ventilation_plan STOP____
ParameterProsition: PCO2 try to install listener
IF-THEN-ELSE: handle_PCO2_plan try Comparison start
ParameterProsition: FiO2 try to install listener
IF-THEN-ELSE: handle_tcSaO2_low_plan try Comparison start
ParameterProsition: FiO2 try to install listener
IF-THEN-ELSE: handle_tcSaO2_high_plan try Comparison start
Return calcpara:PCO2_calculated value 90.0
```

Figure 5: AsbruRTM's Status Output

6 Case Study in NICU

6.1 The Medical Scenario

The primary goal of artificial ventilation is to support breathing until the patient's respiratory efforts are sufficient. Ventilation may be required during immediate care of the newborn who has the respiratory distress syndrome (I-RDS).
In this paper we focus on the controlled ventilation. The intention of the controlled ventilation is to maintain a normal level of the blood values (namely $tcSaO_2$ and $PtcCO_2$) and thus provide the best possible gas exchange by minimal or no lung injury [6]. In Table 2 are listen all needed parameters for the controlled ventilation.

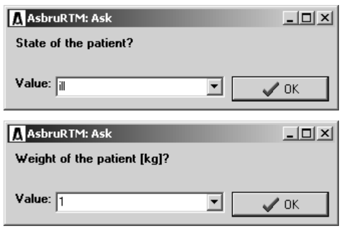
Parameter	In/Out	Description
$tcSaO_2$	In	arterial oxygen saturation
$PtcCO_2$	In	transcutaneous CO_2 pressure
PEEP	Out	positive end-expiratory pressure
PIP	Out	positive inspiratory pressure
FIO_2	Out	fraction of inspired oxygen
f	Out	ventilation frequency

Table 2: Parameters Controlled Ventilation

6.2 Controlled Ventilation in Asbru

The application was tested through a case study. Therefore, an Asbru Light+ plan for the controlled ventilation of newborns was developed (see Figure 8). The evaluation was made "offline", but with the real patient data of newborns. Figure 7 shows the development of the parameters and the appropriate therapeutic recommendations:

- **Init:** First the medical staff enters status and weight of the patient (see Figure 6), then AsbruRTM sets the variables on init values according to Table 3.



AsbruRTM: Ask
State of the patient?
Value: ill
OK

AsbruRTM: Ask
Weight of the patient [kg]?
Value: 1
OK

Figure 6: Evaluation: Initialisation

Status	O_2	PIP	f	PEEP
healty	1	10	40	3
slight ill	30-40	15	40	3-4
ill	40-60	20	40-60	5
severe ill	100	20-25	60	5

Table 3: Initial values Controlled Ventilation

- **Stabilisation of $tcSaO_2$:** The arterial oxygen saturation is too low, FiO_2 is increased every 5 seconds. After 5 minutes the oxygen saturation reaches the target area (see Table 4) and thus FiO_2 must not be increased more. After 8:40 minutes the saturation exceeds the target area and therefore FiO_2 must be diminished.

Parameter	**min**	**max**
$tcSaO_2$	90	92
$PtcCO_2$	40	60

Table 4: Target values Controlled Ventilation

- **Stabilisation of $PtcCO_2$:** The transcutaneous carbon dioxide pressure is too high (see Table 4), hence the ventilation frequency is adapted according to formula 1.

$$f_{target} = f_{actual} * \frac{pCO_2actual}{pCO_2target} \qquad (1)$$

- **PEEP and PIP:** They are hold at level as they are changed not until second instance (when FiO_2 and f cannot stabilise it by there own).

7 Conclusion

We have presented the needs and demands of guideline- and protocol-based care in NICUs and which kinds of functionality are requested. This illustration resulted in the presentation of the powerful and NICU tailored intention-based and time-oriented plan-representation language Asbru. According to Asbru, we have developed an execution engine called AsbruRTM.

Asbru places particular emphasis on an expressive representation of time-oriented actions and states in combination with the underlying intentions. It supports the use of different granularities and reference points to represent multiple time lines. Asbru's representation includes annotations for the duration of actions, their success or failure, and allows time annotation of events, actions/plans, and states with uncertainty in their occurrence. Asbru allows the definition of optional plans as well as mutual exclusive alternatives. It provides a set of temporal relations between sub-plans which exceeds parallel, sequential, any order, and cyclical execution. Such a complex plan-specification language is needed to capture all requirements of a dynamically changing environment, such as in NICUs.

In this paper, we described Asbru's execution capabilities in dynamically changing environments. Asbru's hierarchical structure and knowledge roles facilitate the acquisition and maintenance of knowledge from domain experts. The monitoring module presented ensures proper synchronization of plan execution with a changing environment. This multi-step

```
[...]
<parameter-group title="raw data blood gas online">
   <parameter-def name="tcSaO2" type="parts">
     <raw-data-def mode="automatic" unit="%" channel-name="tcSaO2"/>
       <sampling-frequency>
         <numerical-constant value="10"/>
       </sampling-frequency>
   </parameter-def>
 [...]
</parameter-group>
[...]
<if-then-else>
  <simple-condition>
    <comparison type="equal">
      <left-hand-side>
        <parameter-ref name="patient-state"/>
      </left-hand-side>
      <right-hand-side>
        <qualitative-constant value="healthy"/>
      </right-hand-side>
    </comparison>
  </simple-condition>
<then-branch>
  <variable-assignment variable="FiO2">
    <numerical-constant value="1"/>
  </variable-assignment>
  <variable-assignment variable="PIP">
    <numerical-constant value="10"/>
  </variable-assignment>
  [...]
</then-branch>
</if-then-else>
[...]
<plan name="controlled_ventilation_plan">
  <plan-body>
    <subplans type="parallel">
      <wait-for>
        <static-plan-pointer plan-name=""/>
      </wait-for>
      <plan-activation>
         <plan-schema name="handle_PCO2_plan"/>
      </plan-activation>
         <plan-activation>
      <plan-schema name="handle_tcSaO2_low_plan"/>
      </plan-activation>
    </subplans>
  </plan-body>
</plan>
[...]
```

Figure 8: Asbru Plan: Controlled Ventilation of Newborns

approach allows smooth reaction to unexpected situations ranging from slight parameter deviations to total mission failure.

In the next steps, we will extend Asbru's execution engine to tackle the whole Asbru language and perform a more in-depth evaluation of the Asbru's execution engine.

8 Acknowledgments

This Asgaard Project is supported by "Fonds zur Förderung der wissenschaftlichen Forschung" (Austrian Science Fund), grant P15467-INF.

References

[1] Borland Software Corporation. Borland Delphi. http://www.borland.com/delphi/index.html.

[2] T. Bosse. An interpreter for clinical guidelines in asbru. Master's thesis, Vrije Universiteit Amsterdam, 2001.

[3] J. Bury, J. Fox, and D. Sutton. The PROforma guideline specification language: Progress and prospects. In B. Heller, M. L"offler, M. Musen,

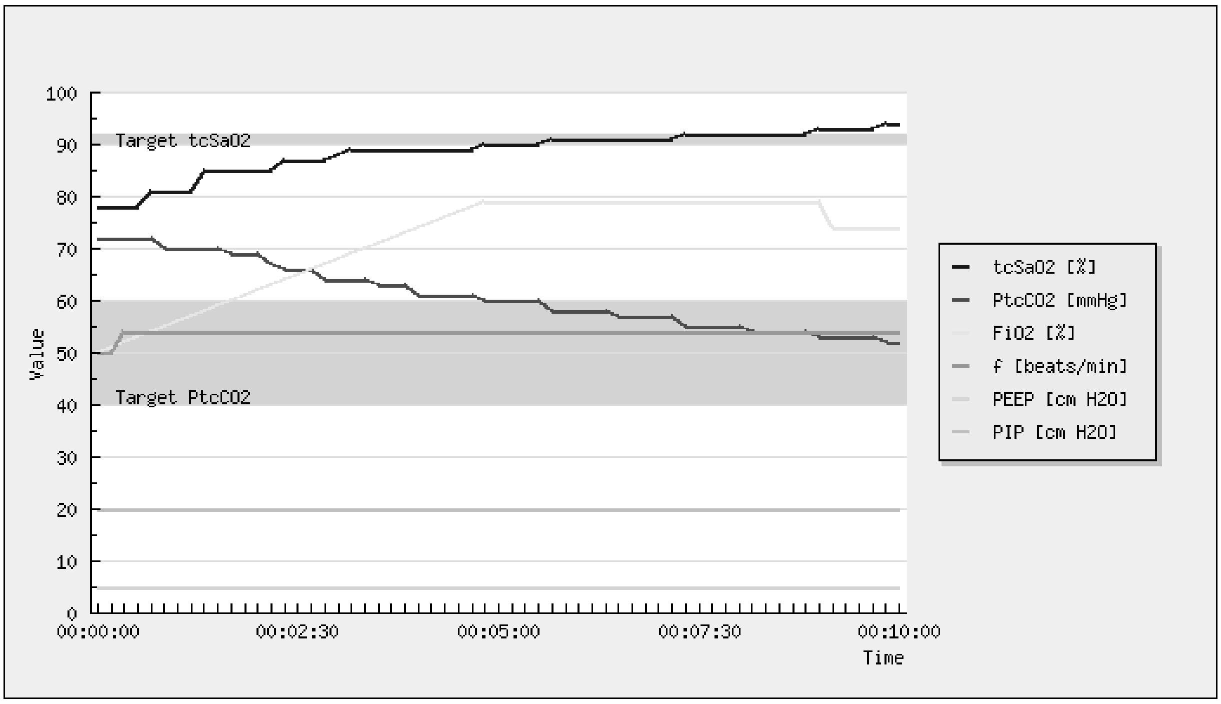


Figure 7: AsbruRTM's Case Study

and M. Stefanelli, editors, *Proceedings of the First European Workshop on Computer-Based Support for Clinical Guidelines and Protocols (EGWLP) 2000*, volume 83:Studies in Health Technology and Informatics, pages 12–29. IOS Press, Amsterdam, 2000.

[4] ExoLab Group. Castor. http://castor.exolab.org.

[5] P. E. Friedland and Y. Iwasaki. The concept and implementaion of skeletal plans. *Journal of Automated Reasoning*, 1(2):161–208, 1985.

[6] Goldsmith and Karotkin. *Assisted Ventilation of the Neonates*. Saunders, Philadelphia, 3 edition, 1993.

[7] P. Johnson, S. Tu, N. Booth, B. Sugden, and I. Purves. Using scenarios in chronic disease management guidelines for primary care. In *Proceddings AMIA Annual Fall Symposium*, pages 389–393, 2000.

[8] S. Miksch. Plan management in the medical domain. *AI Communications*, 12(4):209–235, 1999.

[9] S. Miksch, Y. Shahar, W. Horn, C. Popow, F. Paky, and P. Johnson. Time-oriented skeletal plans: Support to design and execution. In *Fourth European Conference on Planning (ECP'97)*. Springer, September 24–26 1997.

[10] M. Peleg, A. Boxwala, O. Ogunyemi, and et al. Glif3: The evolution of a guideline representation format. In *Proceddings AMIA Annual Fall Symposium*, pages 645–649, 2000.

[11] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and Stefanelli. Comparing computer-interpretable guideline models: A case-study approach. *The Journal of the American Medical Informatics Association (JAMIA)*, 10(1):52–68, 2003.

[12] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible guideline-based patient careflow systems. *Artificial Intelligence in Medicine*, 22:65–80, 2001.

[13] A. Seyfang, R. Kosara, and S. Miksch. Asbru 7.3 Reference Manual. Technical Report Asgaard-TR-2002-1, Vienna University of Technology, Institut of Software Technology & Interactive Systems, Vienna, Austria, 2002.

[14] Y. Shahar, S. Miksch, and P. Johnson. The asgaard project: A task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial Intelligence in Medicine*, 14:29–51, 1998.

[15] S. W. Tu and M. A. Musen. From guideline modeling to guideline execution: Defining guideline-based decision-support services. In *AMIA Annual Symposium*, pages 863–867, Los Angeles, CA, 2000.