Integrated Design and Process Technology, **IDPT-2002**
Printed in the United States of America, June, 2002
©2002 Society for Desing and Process Science

Tools for Acquiring Clinical Guidelines in Asbru

Robert Kosara Silvia Miksch Andreas Seyfang Peter Votruba
Institute of Software Technology and Interactive Systems
Vienna University of Technology, Vienna, Austria
{rkosara,silvia,seyfang}@asgaard.tuwien.ac.at, e9625117@student.tuwien.ac.at

ABSTRACT: In order for clinical guidelines to be verified, they must first be acquired or at least translated into a format that can be treated formally. Most guidelines today either exist as plain text, tables, or flow-charts.

We present two tools that support this translation: The Guideline Markup Tool (GMT) and the Pontifex Intelligent XML Editor Extension (PIXEE). The GMT provides a relatively easy way to translate free text into Asbru. It does this by displaying both the original text and the translation, and showing the user which parts of the Asbru code correspond to which elements of the original text. This not only makes it easier to author plans, but also to understand the resulting Asbru constructs in terms of the original guideline.

PIXEE is a more general XML editor that has some special features due to a richer representation of the language than pure XML. It provides means to aggregate information dynamically and also to more effectively work with language constructs.

Both these tools make the translation into a formal language easier and therefore enable us to formally verify guidelines, thus reducing errors and ambiguities in them.

I. Introduction

Protocol-based care is getting more important as quality assurance and financial constraints are becoming stronger issues in health care [16]. Clinical guidelines so far are mostly written in plain text, decision tables, or flow-charts. This makes them easier to read for physicians without training in formal languages, but also has a number of disadvantages. One of them is that they are often incomplete and ambiguous, and lack any formal means of verification.

In the Protocure [12] project, we want to use formal methods for "curing" protocols from such problems as ambiguity, inconsistencies, etc. But in order to do this, we need a formal representation. Asbru (see Section II) is the language used for Protocure. It provides a very complex set of operators and constructs that effectively represent guidelines.

Translating clinical protocols to formal guideline representations is not an easy task, however [6]. The text guidelines must be adhered to, while at the same time adding information from the common knowledge of physicians (which protocols can naturally rely on). This translation will be done by knowledge engineers, who are also the targeted users of the programs described in this paper.

We need tools that support this task. The first one must not only make it possible to add elements to an Asbru guideline, but also show the user which parts of the original guideline that information comes from. This makes it possible for the physician to understand decisions made by the execution unit (which monitors the patient and gives advice based on the current plan) in terms of the original guideline. The second is a general XML editor which provides some features based on a richer language description than a DTD.

This paper is organized as follows: The next section briefly introduces the Asbru language; Section III lists some related work for our tools; Section IV describes the Guideline Markup Tool and shows how it can be used; Section V introduces the PIXEE XML editor; Section VI describes a scenario to demonstrate the use of the presented tools. We end up with conclusions in Section VII.

II. The Asbru Language

Asbru [8], [9], [13] is a plan representation language that is used in the Asgaard Project[1] to represent clinical guidelines as time-oriented, skeletal plans. It can be used to express clinical protocols as skeletal plans [2] that can be instantiated for every patient.

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

A. Preferences

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

B. 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.

[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/`).

C. *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.

D. *Effects*

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

E. *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 (Table I): in sequence, in any order, in parallel, periodically (as long as a condition holds, a maximum number of times, and with a minimum interval between retries), or unordered (i.e., with no specified constraints).

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.

F. *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.

III. Related Work

Different methods for acquiring knowledge have been proposed. Most of them concentrated on the authoring and ignored the fact that other materials were available. One such example is PROTÉGÉ [10], which generates a structured editor from a language definition. This would appear to be useful for XML applications, but proved very hard to use due to the rigid structure of the generated dialogs.

Another approach was AsbruView [5], a visualization and user interface for editing Asbru plans. The use of metaphors (running tracks, traffic signs) and graphical depiction made the language more accessible to physicians. But this system also ignored all existing materials.

An existing tool similar to the GMT presented here has been proposed for the Guideline Elements Model (GEM [14]): the GEM Cutter [11]. It only allows copying text pieces from the original to the GEM representation, but does not retain the connection to the source position.

HGML [3] is another approach that uses XML for representing guidelines. Its expressive power is, however, much smaller than that of Asbru, and therefore is not suited for formal verification. A somewhat similar approach is Guide-X [15], which is based on XHTML and provides means to markup concepts in the original guideline.

Other guideline modeling techniques have also been proposed [7], [16], but a discussion of them is outside of the scope of this paper.

IV. The Guideline Markup Tool (GMT)

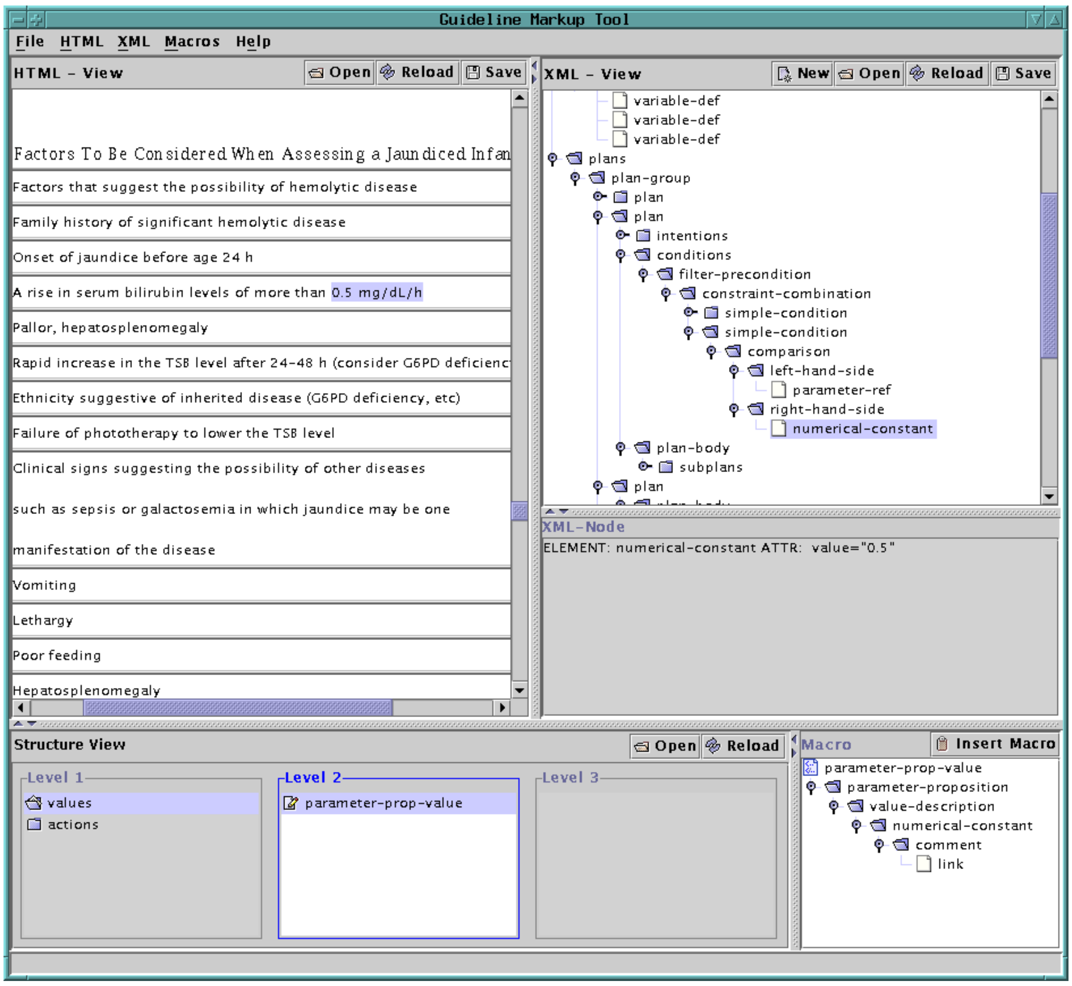
The Guideline Markup Tool (GMT) is an editor that helps translating guidelines from free text into Asbru. It has to support the following tasks:
Authoring Guidelines. We want to be able to take a new guideline in plain text and create an Asbru version of it.
Understanding Guidelines. For an Asbru guideline, we want to be able to see where values in the different parts of the Asbru code come from, and how parts of the original text were translated into Asbru. This is important not just for knowledge engineers, but also for physicians wanting to get an understanding of the language.
Structuring Asbru. The GMT provides a structured list of Asbru elements that needs to be done in a way that best supports the authoring of plans. This list will also provide a good starting point for teaching material and possible subsets of the language for special purposes.

What the GMT does not provide is complex XML editing, because we concentrated on the new interactions instead of implementing a whole editor. The Asbru code that results from the first conversion step will need to be edited further in an XML editor (like PIXEE, which is described below). This second step involves more knowledge of Asbru and its capabilities than medical knowledge – in contrast to the first step. So the users of this program will be physicians (rather than computer scientists), who understand the original guideline, and who can select the relevant parts and associate them with key Asbru parts (a basic knowledge of Asbru is also required, of course).

The GMT is written in Java because of its higher platform independence, clean language design, and freely available user interface and XML components.

	All plans must complete to continue	**Some plans must complete to continue**
Execute in total order (sequence)	*Do-All-Sequentially* Plans (no continuation-condition, all plans must complete)	*Some-Sequentially* Plans (continuation-condition specified as subset of plans)
Start together	*Do-All-Together* Plans (no continuation-condition, all plans must complete)	*Some-Together* Plans (continuation-condition specified as subset of plans)
Execute in any order	*Do-All-Any-Order* Plans (no continuation-condition, all plans must complete)	*Some-Any-Order* Plans (continuation-condition specified as subset of plans)

TABLE I
PLAN TYPES IN ASBRU.



Factors To Be Considered When Assessing a Jaundiced Infan
Factors that suggest the possibility of hemolytic disease
Family history of significant hemolytic disease
Onset of jaundice before age 24 h
A rise in serum bilirubin levels of more than 0.5 mg/dL/h
Pallor, hepatosplenomegaly
Rapid increase in the TSB level after 24–48 h (consider G6PD deficienc
Ethnicity suggestive of inherited disease (G6PD deficiency, etc)
Failure of phototherapy to lower the TSB level
Clinical signs suggesting the possibility of other diseases
such as sepsis or galactosemia in which jaundice may be one
manifestation of the disease
Vomiting
Lethargy
Poor feeding
Hepatosplenomegaly

```
      variable-def
      variable-def
      variable-def
plans
  plan-group
    plan
    plan
      intentions
      conditions
        filter-precondition
          constraint-combination
            simple-condition
            simple-condition
              comparison
                left-hand-side
                  parameter-ref
                right-hand-side
                  numerical-constant
      plan-body
        subplans
    plan
```

ELEMENT: numerical-constant ATTR: value="0.5"

values
actions

parameter-prop-value

```
parameter-prop-value
parameter-proposition
  value-description
    numerical-constant
      comment
        link
```

Fig. 1. The Guideline Markup Tool (prototype). The original plain text guideline is on the left, the corresponding Asbru tree on the right, and the Asbru structure on the bottom. The marked parts of the original guideline and the XML representation are linked.

The GMT window consists of three parts: The left side (HTML View) simply contains the original guideline, where the user can select parts to be connected to Asbru elements later. The XML View on the right side shows the Asbru XML tree, with the lower part giving a more detailed view of the currently selected element (with all its attributes).

The bottom part (Structure View) contains all Asbru elements as well as macros that the user can insert. These elements are organized in a tree, which is represented by the list boxes: each box shows the contents of the node selected in the box to its left. Once a leaf is selected, its contents (the macro or single tag) is shown in the "Macro" window on the bottom right.

In a next step, we will also include a search facility so that the user can find macros or elements by looking for keywords in descriptions that can be added to each macro or element. The will be displayed in their normal position in the tree, so that the user can learn where to look for similar pieces of Asbru code.

A. Interaction

There are two different interactions in the GMT: Adding information to a plan, and browsing a plan.

When writing a new plan or editing an existing one, the user selects a part of the plain text that will be put into the plan. He or she then chooses the appropriate Asbru element or macro from the Structure View, and inserts it into the XML tree by pressing the "Insert Macro" button. The element is inserted as a child of the currently active node, and an Asbru comment with the link to the selected text is added automatically. An HTML anchor is also inserted into the text, which spans the selection and is then used to properly identify the position of the link.

When browsing a plan, the user can select a comment inside any element, and is then shown the corresponding part of the plan. There is also the reverse mapping, which makes it possible to select any part of the plain text guideline and be shown the parts of the plan that point to it (if any).

B. Aliases and Abstract Elements

Pontifex also allows a simple form of inheritance by defining abstract elements that do not appear in the language themselves, but instead one of their children must appear (similar to abstract classes in object-oriented programming, which cannot be instantiated, but a pointer to an object of such a type points to an instance of a sub-class).

Another Pontifex feature are aliases, which allow the language designer to avoid redundancy when defining many tags that are structurally the same, but have different meanings. In such a case, additional names can be assigned to the structure that cause it to be replicated on the XML level.

PIXEE supports abstract types directly, so that the user can see which abstract type would go into a certain position. This is an aid in understanding the structure of the language, because it allows a grouping of connected tags.

C. Configuration

The connection between the Asbru file and the original guideline is made by means of a comment tag that can be put almost anywhere in Asbru, and that can contain a URL. When a part of the original text is selected and a connection made, an HTML anchor is inserted into the text, that the URL in the comment can point to. This anchor does not alter the appearance or contents of the original file in any problematic way.

What element is used to store the link, and what XML has to be inserted to create valid Asbru is specified in a configuration file. Thus, the GMT is not limited to Asbru, and not limited to translating clinical guidelines to Asbru - it could be used for any similar translation.

All elements in the Structure View are also read from the configuration file. The user can insert either single elements or macros. A macro consists of an arbitrary Asbru construct that is inserted instead of a single tag. This makes it possible to provide templates for combinations that are used often, and thus make work faster.

D. Asbru Structure

For this program to be useful in practice, the more than 120 Asbru tags [13] must be structured in a way that makes it possible for the user to easily find the parts he or she needs when translating a plan. For structures that are used quite often, we also want to provide macros or templates that require less work from the user. But these also must be organized in a way that is easy to navigate and use.

We therefore see it as the next challenge to create such a structuring of Asbru that is not oriented at formal criteria, but on the needs of the user when authoring a plan. We also believe that such a structure could prove useful for writing educational material on our language.

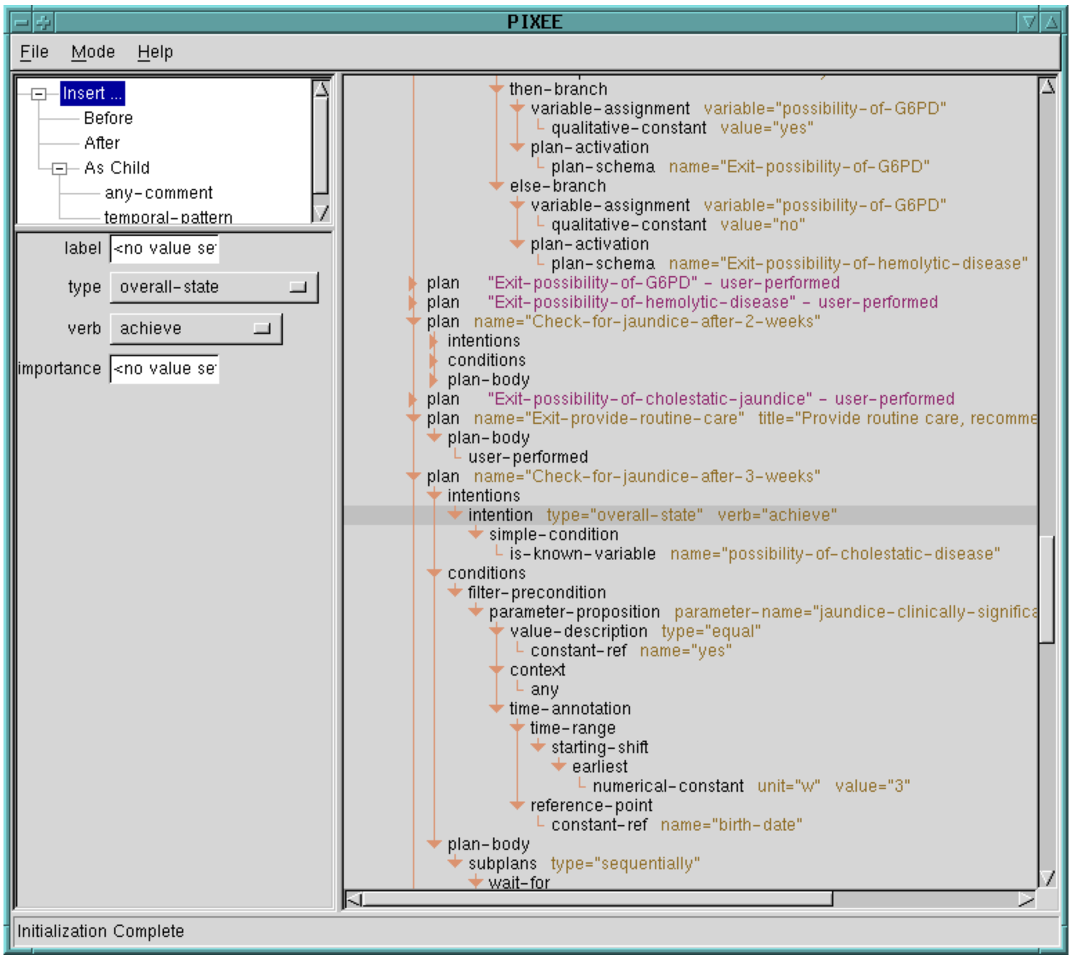
V. PIXEE

PIXEE is an XML editor built on top of *Pontifex* [4] ("bridge builder" – hence the name "Pontifex Intelligent XML Editor Extension"), which is a tool that generates a parser, a class hierarchy and documentation from a language specification of an XML language. In addition to the types of the DTD (document type description), it also has support for types in attributes (like integers, floats, etc), namespaces in ID and IDREF attributes, and a simple form of inheritance. XML Schema [17] is of course much more powerful, but the means of Pontifex were sufficient for the needs of Asbru before Schema became an official recommendation.

PIXEE can be used completely independently from Asbru, of course, but in the context of this project will be used by computer scientists (and possibly physicians with a very good knowledge of Asbru) to change the structure of previously defined parts (e.g., with the Guideline Markup Tool), combine them, add declarations, etc.

Its window (Figure 2) consists of three parts: the control panel on the left which itself is divided into the upper element action part, and the lower attribute editor; and the right side, which shows the XML tree.

PIXEE takes the information about the language from an HSL (Harmless Specification Language) file which is also



PIXEE

File Mode Help

```
Insert ...
  Before
  After
  As Child
    any-comment
    temporal-pattern
```

label <no value se

type overall-state

verb achieve

importance <no value se

```
then-branch
  variable-assignment  variable="possibility-of-G6PD"
    qualitative-constant  value="yes"
  plan-activation
    plan-schema  name="Exit-possibility-of-G6PD"
else-branch
  variable-assignment  variable="possibility-of-G6PD"
    qualitative-constant  value="no"
  plan-activation
    plan-schema  name="Exit-possibility-of-hemolytic-disease"
plan  "Exit-possibility-of-G6PD" - user-performed
plan  "Exit-possibility-of-hemolytic-disease" - user-performed
plan  name="Check-for-jaundice-after-2-weeks"
  intentions
  conditions
  plan-body
plan  "Exit-possibility-of-cholestatic-jaundice" - user-performed
plan  name="Exit-provide-routine-care"  title="Provide routine care, recomme
  plan-body
    user-performed
plan  name="Check-for-jaundice-after-3-weeks"
  intentions
    intention  type="overall-state"  verb="achieve"
      simple-condition
        is-known-variable  name="possibility-of-cholestatic-disease"
  conditions
    filter-precondition
      parameter-proposition  parameter-name="jaundice-clinically-significa
        value-description  type="equal"
          constant-ref  name="yes"
        context
          any
        time-annotation
          time-range
            starting-shift
              earliest
                numerical-constant  unit="w"  value="3"
            reference-point
              constant-ref  name="birth-date"
  plan-body
    subplans  type="sequentially"
      wait-for
```

Initialization Complete

Fig. 2. A screenshot of PIXEE

used for generating the parser etc. by Pontifex [4]. Additional configuration, like the XPath statements associated with certain tags, are read from an additional configuration file which can be user-specific.

A. *Data Aggregation*

When using XML, its tree structure is of great value because the user can select how much and which information to see in an editor. But when hiding a sub-tree, a some information is lost that might still be of interest, even though most of that information is not. In such a case, it would be desirable to be able to extract some of that information and display it even though the subtree is collapsed.

PIXEE provides a means for this by allowing the user to specify XPath and XSL expressions for each tag that are executed when a sub-tree with that tag as the root is collapsed.

Folding and unfolding of elements is done by clicking on the triangles in front of the element's name. When an element is folded, PIXEE runs the XPath query associated with it and displays its results next to the element's name. In Asbru, this can be used to see which plans are user-performed, for example (Figure 2).

B. *IDs and IDREFs*

One key problem when using XML for Asbru was that all ID attributes share one namespace. This problem was overcome with Pontifex, which changed all IDs to simple attributes at the XML level, but managed the namespaces in the parser it generated. This way, any XML tool can still be used for editing Asbru, but of course this is less convenient.

PIXEE knows the namespaces Pontifex used when generating the DTD, and therefore only shows those IDs to the user that are in the namespace of a particular IDREF when edited. It also allows the user to jump from an IDREF attribute to the referenced tag by simply clicking the attribute. This way, navigation between parts of the document is easy.

C. *Attribute Types*

PIXEE not only knows the attribute types of XML, but also the extended ones of Pontifex. It therefore is able to show a list of all possible values for an enumeration attribute, or only accept valid numbers for float fields, etc.

Depending on the type, PIXEE only allows certain inputs for attributes. For enumeration attributes, it displays a list-box of all possible values (Figure 2); this is also done in a similar way for reference attributes: Only the names that are in that namespace are shown and can be inserted.

VI. Using the Tools in Practice

We envision the use of the presented tools similar to the scenario described in the following.

As an example (also for the screenshots), we use the Jaundice guideline (Figure 1) published by the American Academy of Pediatrics [1], [6].

Jaundice is a common disease in new-born babies which is due to an increased bilirubin level (bilirubin is a by-product of hemoglobin breakdown). It is treated by changing the child's food, subjecting him or her to infrared radiation, and possibly other measures, if the disease turns out to be serious (which it is in some cases).

First, the original HTML version of the guideline is loaded into the GMT. The knowledge engineer starts working on a translation from it to the Asbru version. During this time, knowledge is also added from a physician who is an expert in the field of the guideline (like a pediatrician in the case of Jaundice).

When the modeling of the key parts is done, PIXEE is used to restructure the plans, change the way information is passed between them, etc. This (rather complex) process also involves returning to the original guideline repeatedly, which is done with the GMT.

Once the guideline is completely formalized, it can be verified. Any errors that show up in the verification step must be traceable to the original guideline. This is also done by looking up the corresponding parts of the guideline using the GMT. It is also possible that errors are due to the translation or the added knowledge from the expert, which can also be found out relatively easily if a part is erroneous that does not correspond to the guideline.

Finally, when the guideline is eventually used in practice and a recommendation is given based on it, it is possible to track the reasons for such a recommendation back to the original text, while getting the support of a tool that monitors patient values and tells the physician what the guideline recommends at every point.

VII. Conclusions and Future Work

We have presented two tools that help translate free-text guidelines into Asbru to make formal verification possible. The Guideline Markup Tool helps with the initial translation, and makes it possible see which parts of the original guideline a part of the Asbru came from. PIXEE is a tool for editing XML that is specially suited for a complex language such as Asbru, and supports the user when dealing with large files where he or she needs to keep an overview and navigate between the different parts easily.

In the future, we will improve these tools further and develop a structuring of the existing Asbru elements that supports the user when authoring Asbru guidelines.

Acknowledgements

This work is part of the Asgaard Project, which is supported by "Fonds zur Förderung der wissenschaftlichen Forschung" (Austrian Science Fund), grant P12797-INF.

The Protocure project is funded by the IST program of the European Commission under contract number IST-2001-33049.

References

[1] American Academy of Pediatrics' Hyperbilirubinemia Guideline. `http://www.aap.org/policy/hyperb.htm`.

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

[3] C. Greg Hagerty, David Pickens, Casimir Kulikowski, and Frank Sonnenberg. HGML: A hypertext guideline markup language. In *Proceedings Annual Meeting of the American Medical Informatics Association (AMIA)*, pages 325–329, 2000.

[4] Robert Kosara, Klaus Hammermüller, and Silvia Miksch. Co-designing XML-based languages and classes with pontifex. Technical Report Asgaard-TR-2000-1, Vienna University of Technology, Institute of Software Technology, Vienna, Austria, 2000.

[5] Robert Kosara and Silvia Miksch. Metaphors of movement: A visualization and user interface for time-oriented, skeletal plans. *Artificial Intelligence in Medicine, Special Issue on Information Visualization in Medicine*, 22(2):111–131, 2001.

[6] Mar Marcos, Geert Berger, Frank van Harmelen, Annette ten Teije, Hugo Roomans, and Silvia Miksch. Using critiquing for improving medical protocols: Harder than it seems. In *Proceedings of European Conference on Artificial Intelligence in Medicine (AIME 2001)*, pages 431–441, Springer, Berlin, 2001.

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

[8] Silvia Miksch, Yuval Shahar, Werner Horn, Christian Popow, Franz Paky, and Peter Johnson. Time-oriented skeletal plans: Support to design and execution. In *Fourth European Conference on Planning (ECP'97)*. Springer, September 24–26 1997.

[9] Silvia Miksch, Yuval Shahar, and Peter Johnson. Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans. In *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*. Milton Keynes, UK, Open University, 1997.

[10] Mark A. Musen, John H. Gennari, Henrik Eriksson, Samson W. Tu, and Aangel R. Puerta. PROTÉGÉ-II: A computer support for development of intelligent systems from libraries of components. In *Proceedings of the 8th World Congress on Medical Informatics (MEDINFO-95)*, pages 766–770, 1995.

[11] Kristi-Anne Polvani, Abha Agrawal, Bryat Karras, Aniruddha Deshpande, and Richard Shiffman. Gem cutter. `http://ycmi.med.yale.edu/GEM/`.

[12] Protocure. `http://www.protocure.org`.

[13] Andreas Seyfang, Robert Kosara, and Silvia Miksch. Asbru's reference manual, Asbru version 7.2. Technical Report Asgaard-TR-2000-3, Vienna University of Technology, Institute of Software Technology, 2000.

[14] Richard N. Shiffman, Bryan T. Karras, Abha Agrawal, Roland Chen, Luis Marenco, and Sujai Nath. GEM: A proposal for a more comprehensive guideline document model using XML. *Journal of the American Medical Informatics Society (JAMIA)*, 7(5):488–498, October 2000.

[15] Vojtěch Svátek, Tomáš Kroupa, and Marek Růžička. Guide-x – a step-by-step, markup-based approach to guideline formalisation. In Barbara Heller, Markus Löffler, Mark Musen, and Mario Stefanelli, editors, *Computer-Based Support for Clinical Guidelines and Protocols, Proceedings of the First European Workshop on Computer-Based Support for Clinical Guidelines and Protocols (EGWLP)*, volume 83 of *Studies in Health Technology and Informatics*, pages 97–114. IOS Press, 2000.

[16] The Appraisal of Guidelines, Research and Evaluation in Europe (AGREE) Collaborative Group. Guideline development in europe – an international comparison. *International Journal of Technology Assessment in Health Care*, 16(4):1039–1049, 2000.

[17] XML Schema. `http://www.w3.org/XML/Schema/`.