Document-Oriented Views of Guideline Knowledge Bases

Samson W. Tu[1], Shantha Condamoor[1], Tim Mather[2],
Richard Hall[2], Neill Jones[3], and Mark A. Musen[1],

[1] Stanford University School of Medicine
Stanford, CA 94305-5479, USA
[2] SCHIN Lt. Newcastle upon Tyne, UK
[3] University of Newcastle, Newcastle upon Tyne, UK

Abstract. A computer-interpretable guideline knowledge base can be a very large network whose information content is difficult for developers and clinicians to comprehend and review. We created a method to annotate a guideline model and use the annotations to export the guideline knowledge base in an XML format that can be transformed into a readable document. We applied this method to knowledge bases developed in three different guideline modeling projects to analyze uses and limitations of this approach. We demonstrate the promise of creating such document-oriented views, but conclude that guideline models and knowledge bases should be constructed with the goal of creating such human-comprehensible views from the beginning.

1 Introduction

In recent years, professional societies, health-maintenance organizations, medical publishers, and government agencies have produced a flood of clinical practice guidelines (CPGs) for the purpose of disseminating evidence-based best practices. Computer-based clinical decision-support systems that provide assistance in clinical settings have been shown to be effective in improving the performance of care providers [1]. To provide computer-based decision support based on CPGs, the medical knowledge in largely narrative CPG documents must be formalized in computer-interpretable models that can be applied to coded patient data to generate patient-specific recommendations or critiques.

Recent literature on the relationship between narrative text and guideline-based decision support focuses on the translation of such documents to computer-interpretable knowledge bases [2-4].

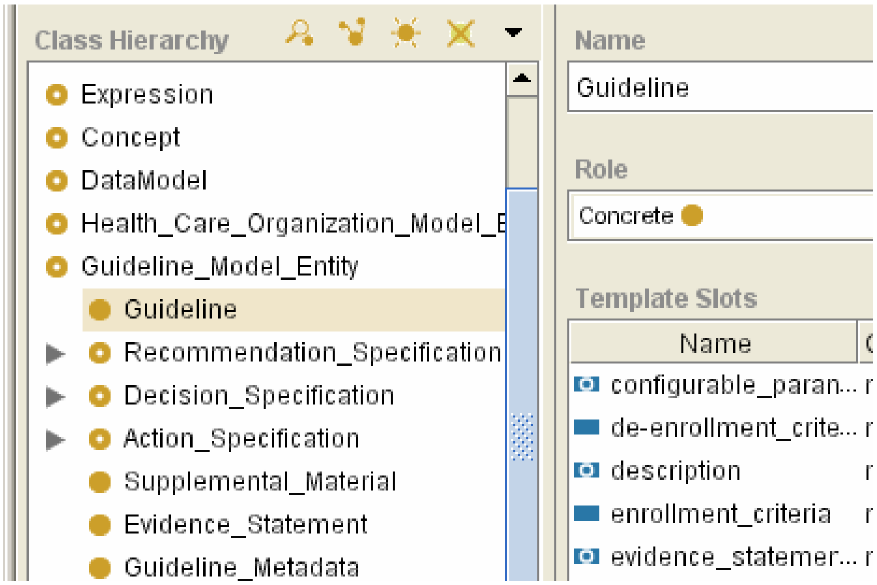
Guideline knowledge bases, however, should be *human-comprehensible* as well as computer-interpretable. Usually, developing such knowledge bases involves collaborations between guideline encoders and subject-matter experts. Without a review format that makes the content of knowledge bases intelligible to those who are not users of sophisticated knowledge-engineering tools, knowledge bases become black boxes that are not subject to human inspection and review. Having such a review format not only benefits collaboration with subject matter experts, but also provides guideline encoders with an efficient method for reviewing a knowledge base systematically, just as code inspection and walkthroughs are part of the software testing process [5]. Furthermore, any sharing of computer-interpretable guidelines across multiple institutions requires that they be reviewable by clinicians.

This paper describes experiments we conducted to create and evaluate document-oriented views of guideline knowledge bases. We developed a method for transforming frame-based knowledge bases to XML and then to HTML. We applied the method to guideline knowledge bases developed in the SAGE [6], ATHENA [7], and PRODIGY[8] projects. We conducted formative evaluation by obtaining feedback from clinical collaborators and by analyzing the document-oriented views in relation to properties of the guideline models used in these projects.

2 Problem Description

The SAGE (Standards-Based **S**harable **A**ctive **G**uideline **E**nvironment) project [6] was a collaboration among research groups at GE Healthcare Integrated IT Solutions, the University of Nebraska Medical Center, Intermountain Health Care, Apelon, Inc., Stanford University, and the Mayo Clinic. The project sought to create the technology for integrating guideline-based decision support into enterprise clinical information systems. The PRODIGY project [8] in the United Kingdom was similarly funded to develop a guideline-based decision-support system that can assist general practitioners in the task of choosing rational therapeutic actions for their patients. The ATHENA Hypertension Guideline Decision Support System [7] was a project that used EON technology [9] to develop and evaluate the implementation of a hypertension guideline decision-support system at Department of Veteran Affairs clinics. All three projects used Protégé-frame [10], a knowledge-engineering environment developed at Stanford University, as the tool for modeling and encoding CPGs. Each project defined its computer-interpretable guideline model as a *guideline ontology* consisting of Protégé classes (Figure 1). Individual guidelines (e.g., a guideline for managing hypertension) were encoded as instances of these classes. Generally, clinicians who were trained in the use of Protégé did the bulk of the encoding work. Each of the CPG knowledge bases typically included several thousand frames. Even expert users of Protégé might have difficulty drilling down to the depth of the knowledge base and understanding the modeling decisions made by the encoders.

The goal of creating a document-oriented view of a knowledge base was to allow subject-matter experts and guideline encoders to read and review the content of the knowledge base systematically. The format should expose large amount of information and should present an accurate view of the computer-interpretable knowledge content. The view should be "readable" on the web or as printed document, where "readable" meant that the generated text was comprehensible to someone not trained in the knowledge representation formalism, and that the document organized significant portions of the content as linear narrative. The document-generation capability should be generic, as we wanted to use the capability to generate views of guideline knowledge bases from different projects. Furthermore, the views should be highly configurable, as custom-tailored presentations might be required for different classes of readers and for different types of content.



Name: Guideline

Role: Concrete ●

Name
configurable_paran...
de-enrollment_crite...
description
enrollment_criteria
evidence_statemer...

Fig. 1. Partial view of the SAGE guideline ontology. The *Guideline* class has properties such as *configurable parameters*, *de-enrollment criteria*, *description*, and *enrollment criteria*.

3 Method

Natural language generation (NLG), a subfield of artificial intelligence, is concerned the generation of text from structured information. In a survey of NLG in healthcare, Cawsey et al. outlined the architecture of NLG as consisting of three stages: (1) text planning (the large-scale organization of the text into coherent sections), (2) sentence planning (the division of information into paragraphs and sentences), and (3) realization (the generation of grammatically correct sentences) [11].

For text planning, we had to determine the scope of information to be presented. A fully developed CPG knowledge base contained not only the content of a computable guideline, but also formal terminologies and a patient data model that were need for the guideline to be implemented in the electronic medical record. We decided that, for our experiments, the scope of the document-oriented view should include frames that were reachable directly and indirectly from instances of the top-level *Guideline* class. In terms of Protégé, it meant the content of the document consisted of trees of instances, where the root of a tree was an instance of the *Guideline* class and branches were the relationships (e.g. *enrollment criteria and recommendation sets*) that defined the structure of a guideline. However, because of interconnectivity of the frames, a simple exhaustive tree-based expansion would result in tremendous repetitions. The content should therefore be partitioned into sections, with hyperlinks connecting references in one section to the content in another. Furthermore, to help navigating the document, we should exploit Protégé graphs that all three projects used to represent task networks of guideline scenarios, decisions, and actions. Thus, the tool should create clickable images of these graphs that allowed a user to navigate to different parts of the document.

For sentence planning, we used an outline format because it corresponded to the underlying nested frame structure and also because scanning an outline was often easier than reading paragraphs. The bottom level of the outline consisted of either text associated with textual properties of a frame or text generated from structured information in one or more frames.

For sentence generation, we decided that, for our initial experiments, instead of finding and using a sophisticated text generator that might add complexity to the system and might not be appropriate for our needs, we would use configurable templates that allowed us to generate the documents quickly.

To support the steps in document generation, we created an *annotation knowledge base* that, for a guideline ontology, specified the large-scale structure of the document and provided context-sensitive templates used by a Java program to convert Protégé instances and graphics into XML fragments and jpeg files (Figure 2). Next, we used Extensible Stylesheet Language Transformation (XSLT), a World Wide Web Consortium standard for rewriting XML documents, to transform the image files and XML file into an HTML file.

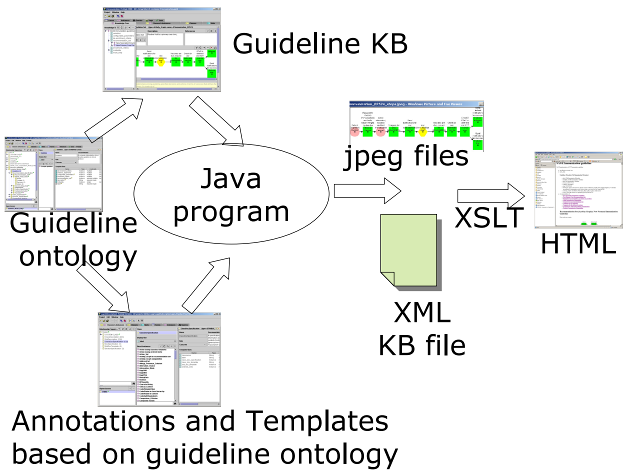


Fig. 2. The process to create a document-oriented view from a guideline knowledge base

The annotation knowledge base for the SAGE guideline ontology specified, for example, the *Guideline, Recommendation,* and *Variables* classes as classes whose instances constituted the main sections of the document to be generated. The Java program then traversed the instance trees anchored by these instances. For each node in the tree, it generated XML output that could be converted to readable text.

To generate the XML output, we created templates for specifying how Protégé instances should be translated. For each Protégé class whose instances we wanted to include in the XML output, we enumerated, as part of the template associated with the class, an ordered list of slots (i.e., properties) whose values should be included in the output. The default XML output used class and slot names as XML tags. Thus, for an instance of the class *Presence_Criterion* in the SAGE guideline model, which had slots *code, presence, valid_window, and vmr_class*, and which allowed a clinician to enter a decision criterion, such as "presence of MMR vaccine administration within last 28 days," in a fill-in-blank GUI form, the XML fragment looks like the following:

```
  <Presence_Criterion p_id= "sageimmunization_02486">
     <code>MMR vaccine</code>
     <presence>true</presence>
     <valid_window>
       <RelativeTimeInteval> …
         </RelativeTimeInterval>
   </valid_window>
    <vmr_class> SubstanceAdministration
   </vmr_class>
</Presence_Criterion>
```

The *RelativeTimeInterval* element expands into an XML fragment that represents an instance of the *RelativeTimeInterval* class (e.g., the interval between 28 days ago and NOW).

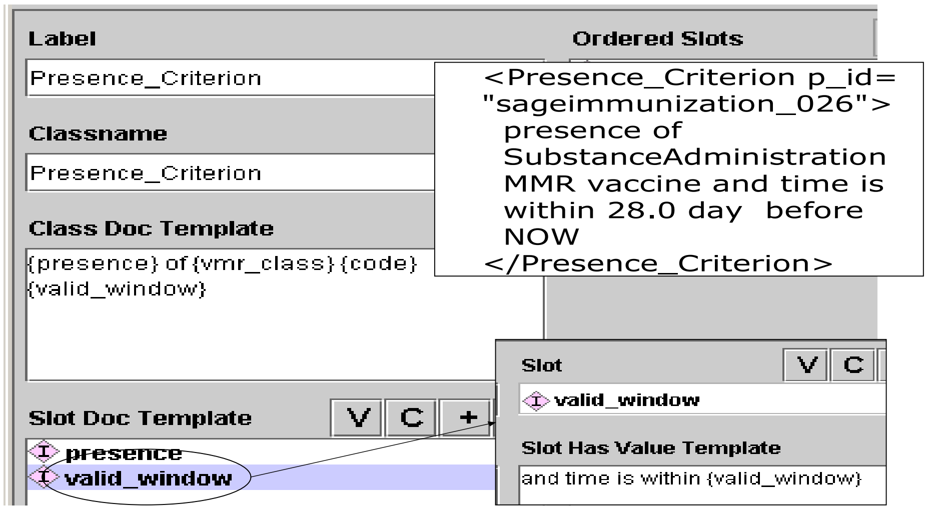
For selected classes in the guideline ontology, we provided alternative templates for specifying textual patterns used to write instances of those classes. The selection of template for an instance was based on the usage context of the instance. Thus, for example, when instances of *Recommendation* referenced instances of *Action* as slot values, one template was used, whereas when instances of class *Decision* referenced instances of *Action*, an alternative template for instances of *Action* was used.

The use of alternative templates for a class allowed us to specify when to expand the content of an instance, and when to reference that instance. In a Protégé knowledge base, frames, such as classes and instances, can be referenced from several other frames. In fact, the reference relationship can be circular. Determining when to expand the content of an instance and when to make a reference to the same content were design decisions that affected the readability of the generated document.

Another reason for providing alternative templates for a class was that we wanted to provide greater user control of the output format. Figure 3 shows an alternative template for generating text for instances of the *Presence_Criterion* class. The pattern "`{presence} of {vmr_class} {code}{valid_window}`" specifies how values of slots should be substituted into the pattern to generate a string. For selected slots, we specified how text could be generated based on presence or absence of slot values. For the slot *valid_window*, the slot template indicates that, the slot value, if it is available, should be preceded by "and time is within " and followed by the expansion of the value of the *valid_time* slot, a *RelativeTimeInterval* instance. The previous XML fragment, using this template, would result in the text "`presence of SubstanceAdministration MMR vaccine and time is within` …" where the elided relative time interval could be "`28 days before NOW`."

4 Results

We wrote scripts to generate default annotation knowledge bases for each of the three guideline ontologies to be tested. Similarly, we developed XSL transforms for each guideline ontology.



{presence} of {vmr_class} {code} {valid_window}

and time is within {valid_window}

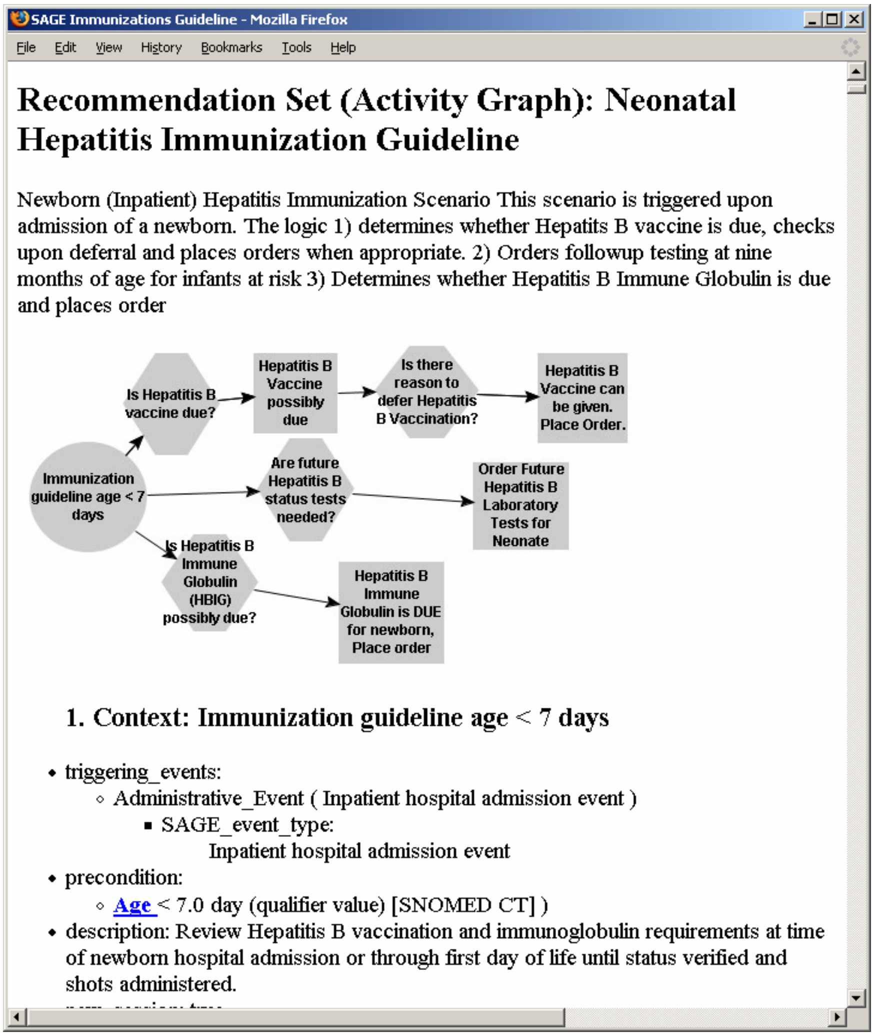
```
<Presence_Criterion p_id=
"sageimmunization_026">
  presence of
  SubstanceAdministration
  MMR vaccine and time is
  within 28.0 day  before
  NOW
</Presence_Criterion>
```

Fig. 3. Alternative template for specifying output format of an instance of Presence_Criterion. The inset box shows an example of the text generated from this template.

Figures 4 and 5 show parts of the HTML pages generated from a SAGE immunization guideline and a PRODIGY post-myocardial infarction guideline. The SAGE HTML page shows the use of graphics and hyperlinks to structure the document. The presentation of the *precondition* of the Context node (an oval in the task graph) shows a formal criterion being displayed as text, with the *Age* variable linked to the section in the document where it is defined.

Because constraints on the resources of the projects involved in this study, a formal evaluation of the document-oriented view of the guideline knowledge base was not feasible. Instead, we performed formative evaluation to explore the text-generation technology's possible uses and limitations, alternative presentations, and properties of guideline knowledge bases that allow better generation of readable text.

The document-oriented view was well-received by the SAGE team. Knowledge-base developers in the project were able to use the HTML document to identify dozens of errors and missing data in the knowledge base. Clinicians found the documents much more accessible than the Protégé knowledge bases from which the documents were generated. However, for the purpose of reviewing the content of the knowledge bases, the clinicians asked for more contextual information about the encoded guideline recommendations. The operationalization of SAGE guidelines involved developing usage scenarios, distillation and interpretation of guideline text, and formalization of decision logic in terms of standard terminologies and a patient information model [6]. Understanding the encoded recommendations required more than having access to the formalized knowledge base. Suggested enhancements to the document included (1) overview documentation to orient a reader, (2) clear indication of relationships between recommendations and sub-recommendations, (3) links to source documents and abstracts of guideline content selected for encoding, and (4) links to example scenarios. Despite these limitations, the document-generation capability proved sufficiently useful for it to be incorporated into the SAGE guideline


SAGE Immunizations Guideline - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Immunization guideline age < 7 days
Is Hepatitis B vaccine due?
Hepatitis B Vaccine possibly due
Is there reason to defer Hepatitis B Vaccination?
Hepatitis B Vaccine can be given. Place Order.
Are future Hepatitis B status tests needed?
Order Future Hepatitis B Laboratory Tests for Neonate
Is Hepatitis B Immune Globulin (HBIG) possibly due?
Hepatitis B Immune Globulin is DUE for newborn, Place order

Recommendation Set (Activity Graph): Neonatal Hepatitis Immunization Guideline

Newborn (Inpatient) Hepatitis Immunization Scenario This scenario is triggered upon admission of a newborn. The logic 1) determines whether Hepatitis B vaccine is due, checks upon deferral and places orders when appropriate. 2) Orders followup testing at nine months of age for infants at risk 3) Determines whether Hepatitis B Immune Globulin is due and places order

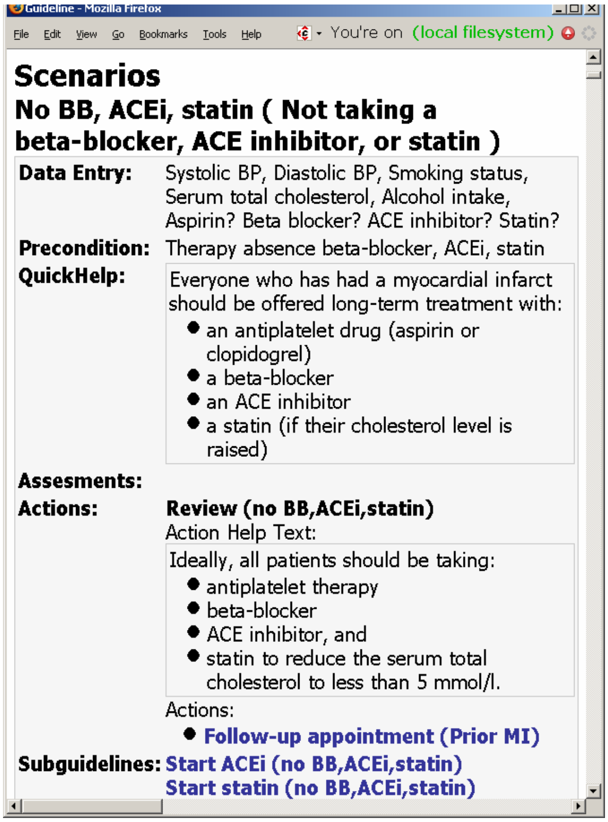
1. Context: Immunization guideline age < 7 days

- triggering_events:
  - Administrative_Event (Inpatient hospital admission event)
    - SAGE_event_type:
      Inpatient hospital admission event
- precondition:
  - Age < 7.0 day (qualifier value) [SNOMED CT])
- description: Review Hepatitis B vaccination and immunoglobulin requirements at time of newborn hospital admission or through first day of life until status verified and shots administered.

Fig. 4. Partial view of the HTML document generated from a SAGE immunization guideline. The clickable image map allows a viewer to navigate to different parts of the document. The circle represents the context of the recommendation (pediatric patient), hexagons decision nodes, and rectangles action nodes.

workbench, thus allowing document-oriented view to be generated at any stage of the knowledge development process.

For the PRODIGY project, the main use case for the document-oriented view involved creating human-readable documents for use by external groups to validate the encoded guideline. Because guideline authors were often far more comfortable authoring in the document paradigm, presenting the complex interconnected network of knowledge components in this way was seen as a significant benefit. The


Guideline - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
You're on (local filesystem)

Scenarios

No BB, ACEi, statin (Not taking a beta-blocker, ACE inhibitor, or statin)

Data Entry:	Systolic BP, Diastolic BP, Smoking status, Serum total cholesterol, Alcohol intake, Aspirin? Beta blocker? ACE inhibitor? Statin?
Precondition:	Therapy absence beta-blocker, ACEi, statin
QuickHelp:	Everyone who has had a myocardial infarct should be offered long-term treatment with: • an antiplatelet drug (aspirin or clopidogrel) • a beta-blocker • an ACE inhibitor • a statin (if their cholesterol level is raised)
Assesments:	
Actions:	**Review (no BB,ACEi,statin)** Action Help Text: Ideally, all patients should be taking: • antiplatelet therapy • beta-blocker • ACE inhibitor, and • statin to reduce the serum total cholesterol to less than 5 mmol/l. Actions: • **Follow-up appointment (Prior MI)**
Subguidelines:	**Start ACEi (no BB,ACEi,statin)** **Start statin (no BB,ACEi,statin)**

Fig. 5. Partial view of the HTML document generated from a PRODIGY post-myocardial infarction guideline. It displays the information associated with a scenario where a patient is not recorded as taking any beta blocker, ACE inhibitor, or statin. The data entry section indicates data that should be acquired in this scenario. The actions associated with this scenario include scheduling follow-up appointment and invoking subguidelines to start ACE inhibitor and statin.

PRODIGY guideline model, as encoded in Protégé, consisted of a series of related projects that allowed the re-use of the reference drugs, clinical terms, and decision criteria. The learning curve for using Protégé exceeded what would be reasonable for external reviewers of encoded guideline content. By flattening out this structure and selectively displaying relevant slots, the document-oriented view provided a convenient method for reviewing and ensuring the quality of the encoded knowledge. One key aspect of the PRODIGY guideline was the concise narrative that was associated with each guideline step and that was displayed to the user when the guideline was executed. By emphasizing this *quick-help* slot in the HTML document (see Figure 5), the generated document allowed an effective way of quality-assuring a large quantity of text without having to access each frame individually. We also recognized the possibility of automatically generating training documentation, and, thus, using the knowledge base for multiple purposes.

The ATHENA hypertension guideline knowledge base had been developed, tested, and deployed over a number of years [7]. A simple-minded hierarchical expansion from the *Guideline* node resulted in an HTML document that contained almost 25 thousand lines. The graph of the clinical algorithm used in the knowledge base proved to be insufficient to allow easy navigation in the generated document. A large part of the hypertension guideline knowledge base dealt with the properties and usage of different classes of anti-hypertensive agents. Alternative presentations (possibly in tabular form) were needed to facilitate easier access to that information than the current hierarchical expansion. Thus, we have not yet presented the HTML document generated from ATHENA to clinicians for external review.

5 Discussion

The idea of generating human-readable documents from machine-interpretable artifacts is not new. Cawsey et al. surveyed several systems in healthcare that used text generation to provide explanations, summaries, reports, and descriptions of medical concepts [11]. The original MYCIN program, for example, included a text-generation module that produced natural-language rule translation [12]. More recently, Design-a-Trial [13] provided a clinical trial authoring environment that generated a protocol document and a Prolog-based executable knowledge base. In the wider computer-science literature, this work on creating document-oriented view of a knowledge base is closely related to Knuth's *literate programming* [14], where the construction of computer programs is seen as a task not only to instruct a computer what to do, but also to explain to human being what we want a computer to do. The experiences we gained from creating document-oriented views of guideline knowledge bases are consistent with the tenets of literate programming.

First, while it important to generate text from the knowledge base, such text does not replace the need to have well-written documentation. The PRODIGY example showed how *quick-help* texts, originally designed as an explanation aid for end users, were helpful to guideline authors for quality-assurance purpose. The SAGE clinician reviewers similarly called for overview narrative to orient readers and to record design decisions. Second, our experiments highlight the importance of mapping knowledge base structures to appropriate document structures. The SAGE guideline knowledge bases, for example, primarily consisted of recommendation sets that provide chapter-like divisions whose content were indexed by graphical algorithms. On the other hand, the ATHENA knowledge base contained heterogeneous knowledge structures that required more complex mapping to a document model.

Just as Knuth designed the WEB system so that a programmer can write documentation and code in the same *literate program* [14], our experiences suggest that computable models of clinical guidelines should be designed so that a knowledge modeler can encode an executable and a readable guideline at the same time. Results in software engineering indicate that the maintenance cost of knowledge bases is likely to exceed their initial development cost. Thus, having human-comprehensible semantics is just as important as having machine-executable formal semantics. Our work demonstrated that, with simple XML output and XSL transformations, it is possible to generate rudimentary documents from multiple guideline knowledge bases. We expect to refine our computer-interpretable guideline models and text-generation capability with the goal of producing better document-oriented views of guideline knowledge bases.

Acknowledgement. This work has been supported in part by grant 70NANB1H3049 of the NIST Advanced Technology Program. We thank Kai Rong at SCHIN for his help in generating the PRODIGY version of the HTML output. The ATHENA project was supported in part by VA HSR&D CPG-97-006, CPI-275, and RCD-96301.

References

1. Hunt, D.L., Haynes, R.B., Hanna, S.E., Smith, K.: Effects of Computer-based Clinical Decision Support Systems on Physician Performance and Patient Outcome: A Systematic Review. JAMA 270(15), 1339–1346 (1998)
2. Shiffman, R.N., Michel, G., Essaihi, A., Thornquist, E.: Bridging the Guideline Implementation Gap: A Systematic, Document-Centered Approach to Guideline Implementation. J Am Med Inform Assoc 11, 418–426 (2004)
3. Shalom, E., Shahar, Y.: A graphical framework for specification of clinical guidelines at multiple representation levels. In: AMIA Annu Symp Proc 2005, pp. 679–683 (2005)
4. Ruzicka, M., Svatek, V.: Mark-up based analysis of narrative guidelines with the Stepper tool. Stud Health Technol Inform 101, 132–136 (2004)
5. Myers, G.J.: The Art of Software Testing. John Wiley & Sons, Inc., Hoboken, NJ (2004)
6. Tu, S.W., Musen, M.A., Shankar, R., et al.: Modeling Guidelines for Integration into Clinical Workflow. Medinfo 174–178 (2004)
7. Goldstein, M.K., Hoffman, B.B., Coleman, R.W., et al.: Implementing Clinical Practice Guidelines While Taking Account of Changing Evidence: ATHENA, an Easily Modifiable Decision-Support System for Management of Hypertension in Primary Care. In: Proc AMIA Symp, pp. 280–284 (2000)
8. Johnson, P.D., Tu, S.W., Booth, N., Sugden, B., Purves, I.N.: Using Scenarios in Chronic Disease Management Guidelines for Primary Care. In: Proc AMIA Symp., pp. 389–393 (2000)
9. Tu, S.W., Musen, M.A.: From Guideline Modeling to Guideline Execution: Defining Guideline-Based Decision-Support Services. In: Proc AMIA Symp., pp. 863–867 (2000)
10. Gennari, J.H., Musen, M.A., Fergerson, R.W., et al.: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. Int J Hum Comput Stud 58(1), 89–123 (2003)
11. Cawsey, A.J., Webber, B.L., Jones, R.B.: Natural language generation in health care. J Am Med Inform Assoc 4(6), 473–482 (1997)
12. Shortliffe, E.H.: Details of the Consultation System. In: Buchanan, B.G., Shortliffe, E.H. (eds.) Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, pp. 78–132. Addison-Wesley Publishing Company, Reading (1984)
13. Modgil, S., Hammond, P.: Decision support tools for clinical trial design. Artificial Intelligence in Medicine 27(2), 181–200 (2003)
14. Knuth, D.E.: Literate Programming. The Computer Journal 27(2), 97–111 (1984)