Modeling Treatment Processes Using Information Extraction*

Katharina Kaiser[1] and Silvia Miksch[1,2]

[1] Vienna University of Technology
Institute of Software Technology & Interactive Systems
Vienna, Austria
http://www.ifs.tuwien.ac.at
{kaiser, silvia}@ifs.tuwien.ac.at

[2] Danube University Krems
Department of Information & Knowledge Engineering
Krems, Austria
http://www.donau-uni.ac.at

Summary. Clinical Practice Guidelines (CPGs) are important means to improve the quality of care by supporting medical staff. Modeling CPGs in a computer-interpretable form is a prerequisite for various computer applications to support their application. However, transforming guidelines in a formal guideline representation is a difficult task. Existing methods and tools demand detailed medical knowledge, knowledge about the formal representations, and a manual modeling.

In this chapter we introduce methods and tools for formalizing CPGs and we propose a methodology to reduce the human effort needed in the translation from original textual guidelines to formalized processable knowledge bases.

The idea of our methodology is to use Information Extraction methods to help in the semi-automation of guideline content formalization of treatment processes. Thereby, the human modeler will be supported by both automating parts of the modeling process and making the modeling process traceable and comprehensible.

Our methodology, called LASSIE, represents a novel method applying a stepwise procedure. The general idea is to use this method to formalize guidelines in any guideline representation language by applying both general steps (i.e., language-independent) and language-specific steps.

In order to evaluate both the methodology and the Information Extraction system, a framework was implemented and applied to several guidelines from the medical subject of otolaryngology. The framework has been applied to formalize the guidelines in the formal Asbru plan representation. Findings in the evaluation indicate that using semi-automatic, stepwise Information Extraction methods are a valuable instrument to formalize CPGs.

* This work is supported by the *Fonds zur Förderung der wissenschaftlichen Forschung – FWF* (Austrian Science Fund), grants P15467-INF and L290-N04.

1 Introduction

Errors in healthcare are a leading cause of death and injury. Kohn et al. [1] mention that, for example, preventable adverse events are a leading cause of death in the United States. In their studies they state that at least 44,000 and perhaps as many as 98,000 Americans die in hospitals each year as a result of medical errors.

Thus, clinical practice guidelines (CPGs) were introduced to increase the quality of care. CPGs are "systematically developed statements to assist practitioners and patient decisions about appropriate healthcare for specific circumstances" [2]. They typically address a specific health condition and provide recommendations to the physician about issues such as who and how to investigate for the problem and how to treat it.

In spite of a substantial level of interest, CPGs have failed to influence clinician behavior significantly. One main reason for this is that guidelines are initially published as textual documents, which require the clinicians to interrupt their workflow to locate, read, and process. If CPGs are embedded within clinical decision support systems (DSSs) integrated within the workflow of clinicians work habits and patient management, they may modify clinicians practices. Thus, to make physicians follow the guidelines, they have to be presented in a structured format that can be used by clinical DSSs.

Therefore, many researchers have proposed frameworks for modeling CPGs in a computer-interpretable and -executable format. Asbru, EON, GLIF, Guide, Prodigy, and PROforma are described by [3] and have been compared by [4]. These frameworks are tailored for specific classes of guidelines, specific users, and specific organizations. Each of these frameworks provides specific guideline representation languages. Most of these language are sufficiently complex that the manual formalization of CPGs is a challenging, but burdensome and time-consuming task. Existing methods and tools to support this task demand detailed medical knowledge, knowledge about the formal representations, and a manual modeling. Furthermore, formalized guideline documents mostly fall far short in terms of readability and understandability for the human domain modeler.

In the next section we describe tools and methods to model computer-interpretable CPGs. In Section 3 we propose our novel methodology to semi-automatically transform process information into a formal representation by a stepwise procedure. We demonstrate the applicability by transforming CPGs into the Asbru representation.

2 Modeling Computer-Interpretable Clinical Practice Guidelines

To support the formalization of CPGs into a guideline representation language various methods and tools exist, ranging from simple editors to sophisticated graphical applications as well as methods that support a step-wise modeling.

2.1 Markup-based Tools

Stepper

Stepper[3] [5] is a mark-up tool for narrative guidelines. The goals of the Stepper project are to develop both a stepwise method for formalization (in this context, XML transformation) of text documents of clinical guidelines and an XML editor enhanced with features to support this method.

Stepper has been designed as a document-centric tool, which takes a guideline text as its starting point and splits the formalization process into multiple user-definable steps, each of which corresponds to an interactive XML transformation. The result of each step is an increasingly formalized version of the source document. An embedded XSLT processor carries out non-interactive transformation. Both the mark-up and the iterative transformation process are carried out by rules expressed in a new transformation language based on XML, the so-called XKBT (XML Knowledge Block Transformation). Stepper's transformation process consists of six steps:

1. *Input text format.* The format of the original guideline text is XHTML, the XML version of HTML.
2. *Coarse-grained semantic mark-up.* Basic blocks of the text are marked (e.g., headings, sentences) and parts without operation semantics are removed.
3. *Fine-grained semantic mark-up.* Complex sentences are rearranged into simpler ones and background knowledge is added. In addition, a data dictionary is created, which describes the clinical parameters involved.
4. *Universal knowledge base.* The original document is transformed into a universal knowledge base. This involves changing the structure of the document to achieve modularity, which is assumed to involve medical experts in part.
5. *Export-specific knowledge base.* The representation is adapted to ease the export to the target representation. Therefore, an export-specific knowledge base is produced from the universal one.
6. *Target computational representation.* The ultimate format is produced by the knowledge engineer. This step is assumed to be performed fully automatically using XSL style sheets.

[3] http://euromise.vse.cz/stepper-en/aplikace_stepper/index.php (Accessed: Sep. 17, 2005)

By using the Stepper method and tool it is possible to transform CPGs into fragments of operational code (e.g., Java) or into parts of a guideline representation language (e.g., Asbru).

Stepper's main advantage is the documentation of all activities, which allows to easily review the transformation process. Stepper also provides an interface showing the interconnection between the source text and the model.

GEM Cutter

The *GEM Cutter* [6] is a tool with the aim to facilitate the transformation of CPGs into the Guideline Elements Model (GEM) format [7], an XML-based guideline document model. The model encodes considerable information about guideline recommendations in addition to the recommendations themselves, including the reason for each recommendation, the quality of evidence that supports it, and the recommendation strength assigned by the developers.

The authoring process for GEM guidelines takes place in three steps:

1. The GEM document, which has an XML-based syntax, is created based on the original guideline using the GEM Cutter (see Figure 1). The elements of the GEM document are then stored in a relational design database.
2. *Knowledge Customization:* meta-information is added, the guideline can be locally adapted, and abstract concepts of the guideline can be implemented. This step is guided by the *knowledge customization wizard*.
3. *Knowledge Integration* into the clinical workflow depending on local circumstances.

The GEM Cutter shows the original guideline document together with the corresponding GEM document and makes it possible to copy text from the guideline to the GEM document. The GEM document is displayed in tree structure format. Each item on the tree represents a GEM element, which can contain text and can be edited.

Document Exploration and Linking Tool/Addons (DELT/A)

DELT/A [9] provides a relatively easy way to translate free text into various (semi-)formal, XML-based representations. It achieves this by displaying both the original text and the translation, and showing the user which parts of the formal code correspond to which elements of the original text. This not only makes it easier to author plans, but also to understand the resulting constructs in terms of the original guideline.

DELT/A[4] provides two main features: (1) linking between a textual guideline and its formal representation, and (2) applying design patterns in the form of macros.

[4] http://ieg.ifs.tuwien.ac.at/projects/delta/ (Accessed: Sep. 9, 2005)

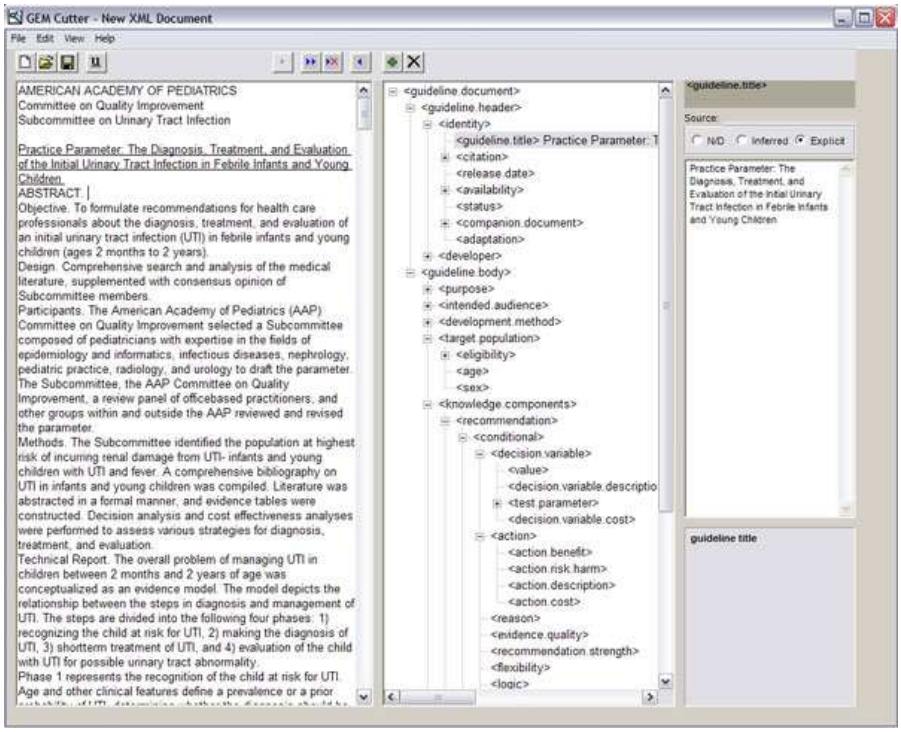


Fig. 1. GEM Cutter [8]. The left pane shows the guideline text, the middle pane shows the GEM tree segment and the right pane shows the Element Segment.

DELT/A allows the definition of links between the original guideline and the target representation, which gives the user the possibility to find out where a certain value in the XML-language notation comes from. If someone wants to know the origin of a specific value in the XML file DELT/A can be used to jump to the correlating point in the text file where the value is defined and the other way round. The second feature is the usage of macros. A macro combines several XML elements, which are usually used together. Thus, using macros allows creating and extending specific XML files more easily through the usage of common design patterns.

DELT/A (see Figure 2 for its user interface) supports the following tasks:

Authoring and augmenting guidelines. The user wants to be able to take a new guideline in plain text and create an (XML-based) representation of it, and to add links to the corresponding parts of a guideline to an already existing XML file.

Understanding the (semi-)formal representation of guidelines. For a guideline in a (semi-)formal representation, the user wants to be able to see where values in the different parts of the representation's code come from, and how parts of the original text were translated into it. This is

important not just for knowledge engineers, but also for physicians, who want to get an understanding of the language.

Structuring the syntax of the (semi-)formal representation. DELT/A provides a structured list of elements of the target language – the macros – that need 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.

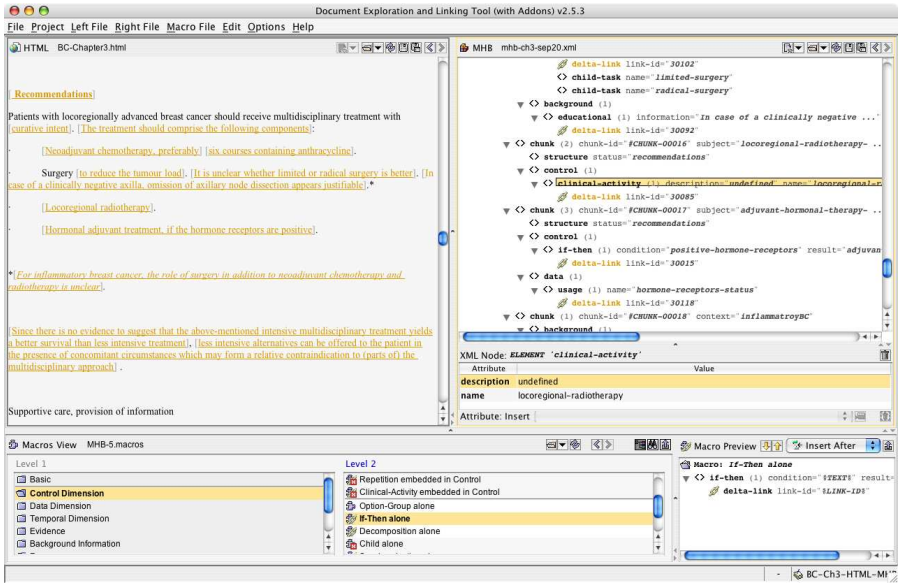


Fig. 2. Document Exploration and Linking Tool / Addons. The left pane shows a guideline document in textual format. The right pane shows a formal representation of the document. The bottom pane shows the Macros that can be used to support the formalization task by templates of several models.

By means of these features the original text parts need not be stored as part of the target representation elements. The links clearly show the source of each element in the target representation. Additionally, there is no need to produce a guideline in natural language from the target representation since the original text remains unaltered.

Uruz/Degel – Digital Electronic Guideline Library

Degel[5] [10] is a generic framework with tools to support guideline classification, semantic markup, context-sensitive search, browsing, run-time application, and retrospective quality assessment. It is applicable for any XML-based guideline representation, currently supporting Asbru and GLIF [3]. It supports the gradual migration of free text guidelines to formal representations.

Semantic markup is performed using the Uruz web-based guideline markup-tool. It can also be used to create a guideline document de-novo (i.e., without using any source) by directly writing into the knowledge roles of a selected target ontology. The editor can modify the contents or add new content. This enables implicit knowledge to become more explicit, further facilitating the task of the knowledge engineer who fully formalizes the guideline.

Several features are especially tailored to Asbru, such as the plan-body wizard (PBW), which is used for defining the guideline's control structure. The PBW enables a user to decompose the actions embodied in the guideline into atomic actions and other sub-guidelines, and to define the control structure relating to them (e.g., sequential, parallel, repeated application). The PBW, used by medical experts, significantly facilitates the final formal specification by the knowledge engineer.

To be truly sharable, guidelines need to be represented in a standardized fashion. Thus, Uruz enables the user to embed in the guideline document terms originating from standard vocabularies, such as ICD-9-CM (International Classification of Diseases) for diagnosis codes, CPT-4 (Current Procedural Terminology) for procedure codes, and LOINC-3 (Logical Observation Ientifiers, Names and Codes) for observations and laboratory tests.

2.2 Graphical Tools

AsbruView

Within the Asgaard/Asbru project a graphical user interface to Asbru [3] was developed, which supports the development of guidelines and protocols, called AsbruView[6] [11].

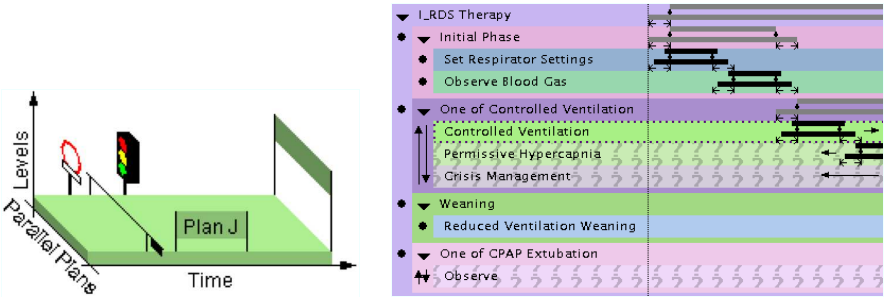
Asbru is a complex language, which cannot be fully understood by physicians, who have no or hardly any training in formal methods. AsbruView is a tool to make Asbru accessible to physicians, and to give any user an overview of a plan hierarchy. AsbruView is based on visual metaphors to make the underlying concepts easier to grasp. This was done because not only is the notation foreign to physicians, but also the underlying concepts. AsbruView

[5] http://medinfo.ise.bgu.ac.il/medlab/ResearchProjects/RP_DeGeLhtm.htm (Accessed: Aug. 25, 2005)

[6] http://www.asgaard.tuwien.ac.at/tools/asbruview.html (Accessed: June 12, 2005)

provides four views: Topological View (see Figure 3(a)), Temporal View (see Figure 3(b)), SOPOView, and XML View.

The metaphors and graphical representation of AsbruView have proved to be useful in communicating Asbru's concepts to physicians. Users get a better overview of the therapy steps than from tables, while at the same time being able to see the precise temporal constraints of plans (which is not the case with flowcharts).



(a) Topological View (b) Temporal View

Fig. 3. AsbruView

Protégé

Protégé[7] is an open source ontology development and knowledge acquisition environment [12]. It is a Java tool, which provides an extensible architecture for the creation of customized knowledge-based tools and assists users in the construction of large electronic knowledge bases. Protégé has an intuitive user interface that enables developers to create and edit domain ontologies and supports customized user-interface extensions, incorporates the Open Knowledge Base Connectivity (OKBC) knowledge model, and interacts with standard storage formats such as relational databases, XML, and RDF.

Protégé is a 'meta-tool' that helps users construct domain-specific knowledge acquisition systems that application experts can use to enter and browse the content knowledge of electronic knowledge bases. It is also a knowledge-base editing tool, which supports the construction of a domain ontology, the design of customized knowledge-acquisition forms, and entering domain knowledge. Furthermore, it serves as a platform, which can be extended with graphical widgets for tables, diagrams, and animation components to access other knowledge-based systems embedded applications. Protégé is a library, which other applications can use to access and display knowledge bases.

[7] http://protege.stanford.edu (Accessed: June 15, 2005)

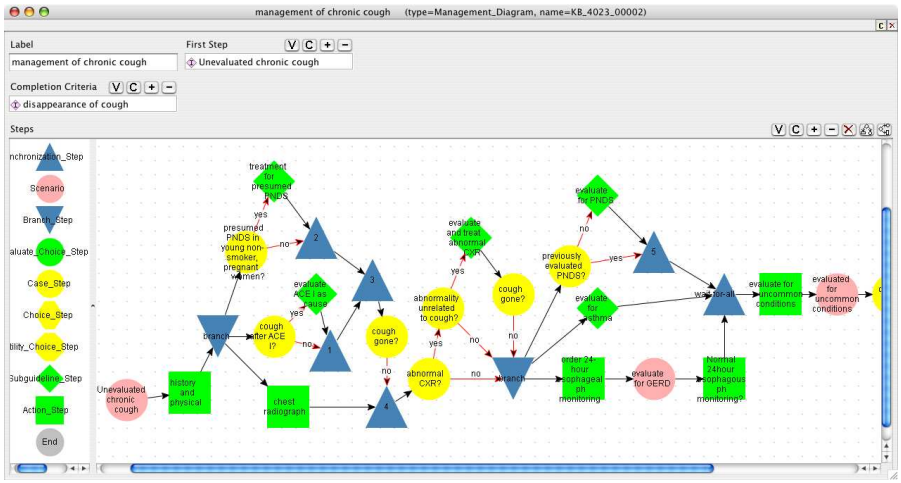


Fig. 4. View of Protégé being used to author a guideline for managing chronic cough. The guideline model being used in this application is Dharma, part of the EON framework.

Protégé is also used to author guidelines in various models (e.g., EON, GLIF, Prodigy, PROforma). Thereby, a part of the modeling can be accomplished using predefined graphical symbols. These symbols are arranged in a diagram and linked by graphs. See Figure 4 for an example. The underlying data is entered by web masks.

Arezzo

The first implementation of software to create, visualize, and enact PROforma [13] guidelines was *Arezzo*. It consists of three main modules: (1) the Composer, (2) the Tester, and (3) the Performer (see Figure 5).

The *Composer* is a graphical editor or knowledge authoring tool, which uses PROforma notation to capture the structure of a guideline and generate an executable specification (see Figure 6 for an overview). The building blocks used to construct a guideline of any level of complexity are the four PROforma task types. Data items and their properties to be collected during protocol enactment are also defined using the Composer.

The *Tester* is used to test the guideline logic before deployment.

The *Performer* executes guidelines defined in the PROforma language. It interprets the guideline specification and during guideline enactment it prompts the user to perform actions, collect data, carry out procedures, and make decisions as required.

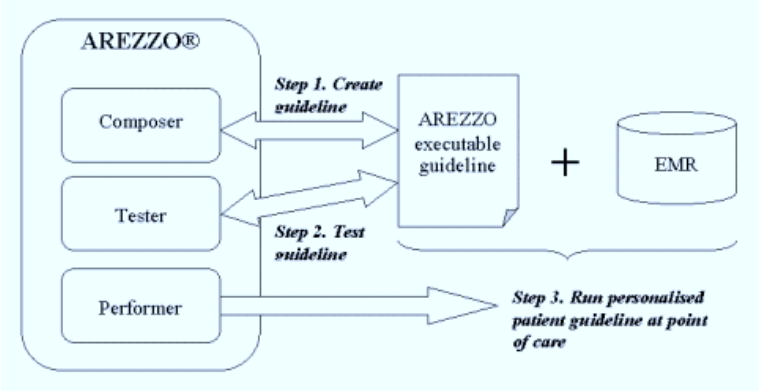


Fig. 5. Overview of the Arezzo application [14].

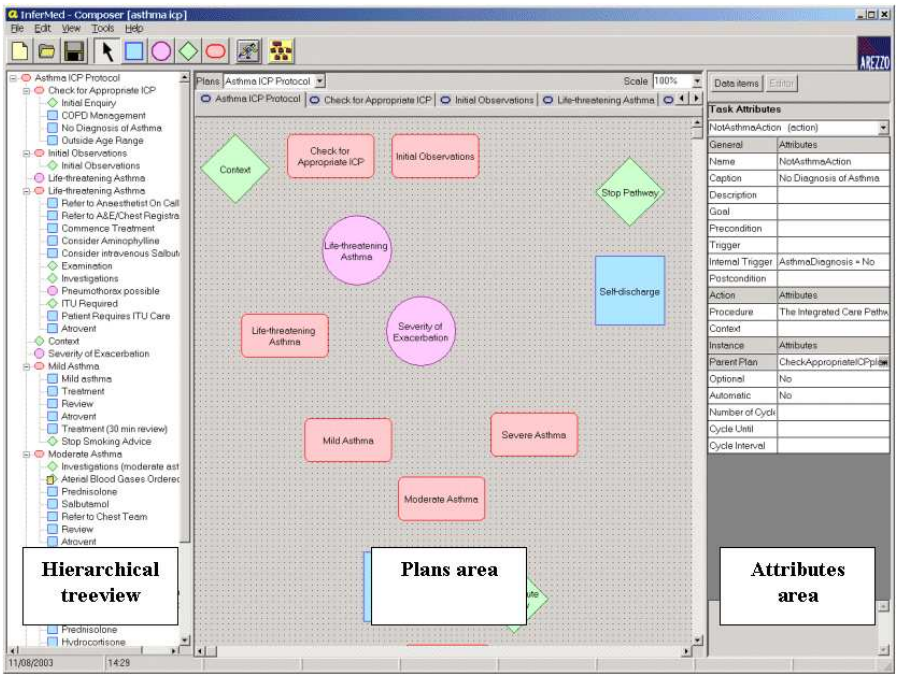


Fig. 6. AREZZO Composer [15]. The three-panel screen, with a hierarchical tree view of the guideline tasks in the left pane, task-authoring tool in the middle pane, and the attributes-authoring tool in the right pane.

Tallis

Tallis [16] is a new Java implementation of PROforma-based authoring and execution tools developed by Cancer Research UK. It is based on a later version of the PROforma language model and consists of a Composer (to support creation, editing, and graphical visualization of guidelines), Tester, and Engine (to enact guidelines and allow them to be manipulated by other applications). Tallis is also designed for delivering web-based services; applications will run on any platform and integrate with other components, including 3rd party applications. The Tallis Publisher forms part of the Tallis software suite. It has been built to allow guidelines to be published and enacted over the WWW.

2.3 Multi-step Methodologies

In the course of time most of the guideline representation languages have produced a very extensive syntax. Due to this complexity the modeling process of a CPG into such a language is a very challenging task. Nowadays, researchers know the requirements to such a language and the resulting complexity. Often, they develop both a particular guideline representation language and methodologies to model guidelines together. Many of these methodologies result in a multi-step approach, as a one-step or even a two-step modeling process was shown to be not sufficient to the modeler [17, 18].

Some tools have already been developed to support multi-step methods, such as *Stepper* [5] or the *GEM Cutter* [6]. But there are also other methodologies that were developed either together with the representation language (e.g., SAGE [19]) or afterwards when it was obvious that the modeling process needed systematically structuring and adaptation for the various groups of modelers, such as knowledge engineers and physicians (e.g., MHB [18, 20]).

SAGE – The Standards-based Shareable Active Guideline Environment

The Standards-based Shareable Active Guideline Environment (SAGE) [19] uses standardized components that allow interoperability of guideline execution elements with standard services provided within vendor clinical information systems. It includes organization knowledge to capture workflow information and resources needed to provide decision-support in enterprise setting. It synthesizes prior guideline modeling work for encoding guideline knowledge needed to provide situation-specific decision support and to maintain linked explanatory resource information for the end-user.

The SAGE methodology for developing a guideline knowledge base consists of six main steps (cp. Figure 7).

1. Clinicians must create clinical scenarios that are detailed enough to support integration of executable guideline content into real clinical workflow.

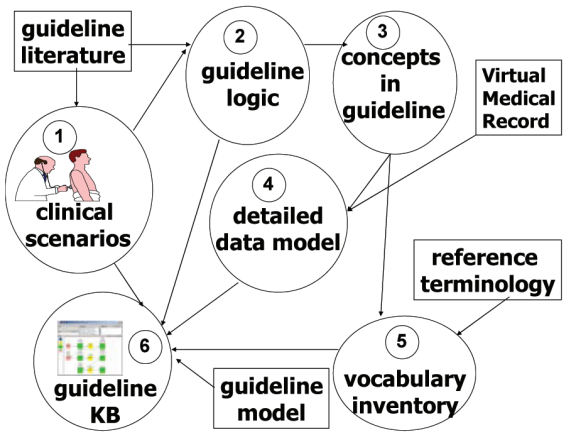


Fig. 7. Steps in modeling clinical practice guidelines for integrating into workflow. The arrows represent information flow [21].

2. Clinicians extract the knowledge and logic needed to generate these recommendations from guideline texts, medical literature, and clinical expertise. The extraction process requires clinicians to select, interpret, augment, and operationalize guideline statements to disambiguate concepts.
3. Clinical concepts used in the extracted guideline logic are identified.
4. Concepts identified as part of the required guideline logic are instantiated as detailed data models that correspond to constraints on classes of "virtual medical record" (vMR). The vMR supports a structured data model for representing information related to individual patients, domains for values of attributes in the data model, and queries through which guideline DSS can test the various patient states.
5. Guideline concepts in terms of standard terminologies are specified. To implement a computerized guideline in a particular institution, terms used in a guideline knowledge base to describe patient states must be mapped to terms in that institutions electronic patient record. Standard terminologies, such as SNOMED CT2[8] (the Systematized Nomenclature of Medicine Clinical Terms) and LOINC-3[9], provide the necessary shared semantics for such mappings.
6. Clinical scenarios and guideline logic are translated into a computer-interpretable model of guidelines. The SAGE methodology calls for explicit modeling of guideline usage as part of the executable guideline specification. As such, it assumes that a guideline does not dictate the workflow in a clinic, but the guideline knowledge base specifies how a DSS reacts to events in the care process.

[8] http://www.snomed.org (Accessed: Sept. 17, 2005)
[9] http://www.regenstrief.org/loinc/ (Accessed: Oct. 4, 2005)

When encoding a guideline for SAGE, clinical experts must interpret the guideline statements and create one ore more plans that will support the guideline goals in the specific work environment of their healthcare organization.

MHB - A Many-headed Bridge between Guideline Formats

Transforming the original guideline text into a formal guideline representation is a difficult task requiring both the skill of a computer scientist and medical knowledge. To bridge this gap, an intermediate representation called MHB [18, 20] has been designed. It is called a *Many-Headed Bridge* between informal representations such as free text and tables and more formal representations such as Asbru, GLIF, or PROforma [3].

The overall structure of an MHB file is a series of *chunks* corresponding to a certain bit of information in the natural language guideline text (i.e., a sentence, part of a sentence, more than one sentence). The information in a chunk is structured in various dimensions. MHB provides eight different dimensions:

1. *Control flow dimension:* It is used to define when to do what. Therefore, two means are offered to express this: (1) *decisions* in the form of 'if-then' statements and (2) *decomposition*, where a task and its subtasks are named.
2. *Data flow dimension:* It consists of the description of the data processing involved in diagnosis and treatment. It is composed of the 'definition' and the 'usage' of the data item.
3. *Temporal aspects dimension:* MHB covers the complexity of Asbru in modeling temporal aspects and is extended by more standard concepts such as average or precise duration. That means that for each start, end, and duration, the minimum, maximum, estimate, and precise value can be given.
4. *Evidence dimension:* Thereby, a 'grade' is given to show the overall strength of the evidence supporting a scientific conclusion and a 'level' is given for every single literature reference that this statement is built on. The level depends on the quality and the design of the study.
5. *Background information dimension:* Information about intentions, effects, relations, and so on can be stated.
6. *Resources:* The resources consumed by an action are described, such as *personal* resources, *devices*, and *financial* costs.
7. *Patient related aspects dimension:* These are other issues which see treatment from the patient perspective, such as *risk*, *patient discomfort*, and *health prospective*.
8. *Structure dimension:* Thereby, the position of the chunk within the guideline document can be stated (i.e., *introduction*, *definition*, *recommendations*, etc.).

MHB not only provides constructs to express the essential knowledge, but also allows for a modeling with the degree of detail necessary for further modeling purposes. When translating an MHB guideline to a guideline representation format such as Asbru, together with the original guideline text, it forms a better basis for guideline formalization than the original guideline text alone.

Experiences [18] have shown that MHB is easier to understand than, for instance, Asbru for those without computer background, although a significant effort in training was necessary. Furthermore, it is easier to create an MHB model than an Asbru model from the original guideline text and it is easier to create an Asbru model based on MHB than based on the original guideline.

3 LASSIE: Semi-automatic Modeling Using Information Extraction

In the preceding section we have shown various tools and methodologies to create a guideline model in a formal guideline representation language. The major drawback for all those tools and methods is that they demand for a burdensome and time-consuming manual modeling. Furthermore, they mostly lack readability/understandability once the guideline document is formalized in the representation language. If the modeling process is carried out in multiple steps, often the traceability of each step is not possible. Some languages provide no or only rudimental support to select the appropriate syntax for modeling a particular part of a guideline. Thus, we propose a new methodology that accomplishes parts of the modeling automatically and provides traceability for each modeling process step.

3.1 Our Approach

Inspired by multi-step modeling methodologies, such as Stepper, SAGE, or GEM, we defined a stepwise approach, which should be applicable for various representation languages (see Figure 8) [22]. It facilitates the formalization process by using various intermediate representations (IRs) that are obtained by stepwise procedures. The IRs are specific templates that can present the desired information. The transformation from one representation to another is applied using Information Extraction (IE) methods (see [23, 24] for surveys).

IRs cover the representation of a particular piece of information (i.e., actions, processes, sequences, parameter definitions, etc.) [22]. They are XML-based documents and are constructed by refining the preceding representation. The stepwise procedure also enables the user to interact, to alter and to improve the output of each step in order to enhance the output of subsequent steps. This is necessary as the whole process of computerization, and especially the first steps of formalization, cannot be fully automated due to the content of original textual guidelines, which might be ambiguous, vague, and

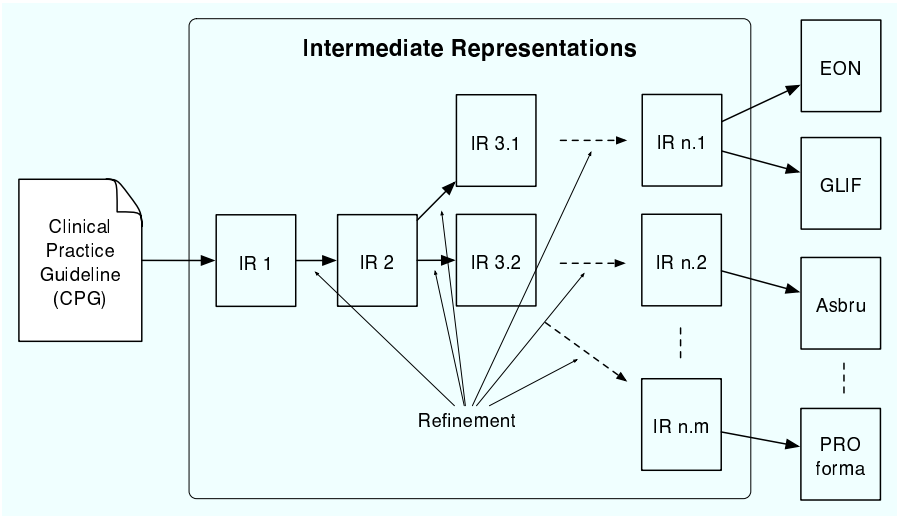


Fig. 8. Idea of the modeling approach. Starting from the guideline document at the left side the stepwise approach using IRs obtains a formalized guideline in a guideline representation language on the right side.

incomplete. Guidelines for intended users of medical personnel do not explicitly describe common knowledge required to accomplish their daily work (e.g., what is 'fever' and at which blood temperature do we define 'high fever'?). If guidelines have to be executed in a computer-supported way it is necessary to model this implicit information. Due to the lack of this knowledge in an explicit form, it is not possible to model this information automatically. In our system we are restricted to the processing of explicitly described information in the guidelines. If additional knowledge is required it has to be added by a domain expert.

To process as large a class as possible of documents and information we need specific heuristics. These are applied to a specific form of information, for instance:

Different kinds of information. Each kind of information (e.g., processes, parameters) needs specific methods for processing. By presenting only one kind of information the application of the associated method is simpler and easier to trace.

Different representations of information. We have to take into account various ways in which the information might be represented (i.e., structured, semi-structured, or free text).

Different kinds of guidelines. CPGs exist for various diseases, various user groups, various purposes, various organizations, and so on, and have been developed by various guideline developers' organizations. Therefore,

we can speak about different classes of CPGs that may contain similar guidelines.

Hereby, we want to emphasize that is not our intention to develop new representations for clinical guidelines. The IRs are only means in our approach to semi-automatically generate a formal representation in any guideline representation language.

3.2 The Methodology

CPGs present effective treatment processes. One challenge when authoring CPGs is the detection of individual processes and their relations and dependencies. We will demonstrate that it is possible to formalize processes using IE and present a framework for modeling guidelines in Asbru [3] as proof of concept (see Figure 9).

Our multi-step transformation methodology, called *LASSIE*[10], facilitates the formalization process by using various IRs that are obtained by stepwise procedures. LASSIE is intended to be a semi-automatic approach. This enables the user not only to correct the transformations, but also to augment them by implicit knowledge necessary for a subsequent execution.

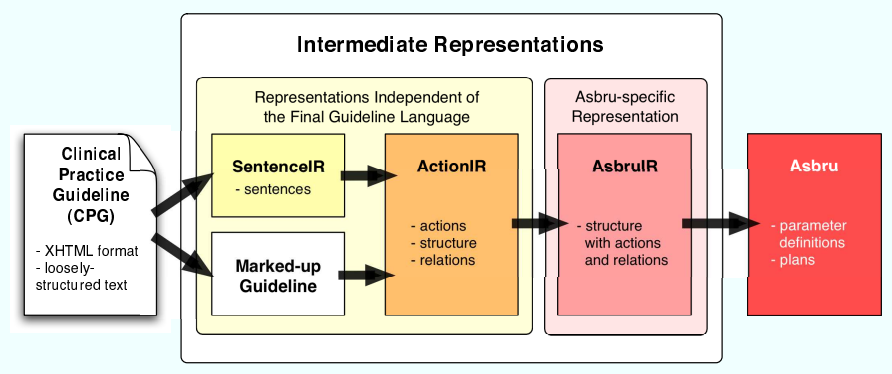


Fig. 9. Steps to (semi-)automatically gain an Asbru representation of CPGs. To gain process information from a CPG the first two steps are accomplished in order to have a representation independent of the final guideline language.

In the remainder of this section we present the steps that extract process information from clinical guidelines using heuristic algorithms. These algorithms are based on extraction patterns whose obtaining is described in this section. The output of this method is a unified format, which can be transformed into the final representation, but which is independent of the final

[10] modeLing treAtment proceSSes using Information Extraction

guideline representation language. In order to create the guideline in the Asbru format we describe how the IR can be transformed into Asbru.

Developing Extraction Rules

Information Extraction (IE) is an emerging Natural Language Processing (NLP) technology whose function is to process unstructured, natural language text, to locate specific pieces of information, or facts in the text, and to use these facts to fill a database [25]. Similar to IE systems are *wrappers* which aim to locate relevant information in semi-structured data [26] and often do not need to apply NLP techniques due to a less grammatical structure of the information resources.

Approaches for developing IE systems

For developing both IE and wrapper systems two approaches can be applied: (1) the Knowledge Engineering approach and (2) the automatic learning approach.

The former is customized manually to a given task (e.g., [27, 28, 29]). But manually generating extraction rules is a cumbersome and time-consuming task. Thus, research has been directed towards automating this task. The automatic approach takes a set of documents and outputs a set of extraction patterns by using Machine Learning techniques. Automatic learning systems can be categorized in three groups:

1. *supervised learning systems*, where a large set of training data is required to learn the rules using Machine Learning techniques (e.g., [30, 31, 32]),
2. *semi-supervised learning systems* (e.g., [33, 34]), and
3. *unsupervised learning systems*, where rules are learned by a small set of seed rules and an annotated corpus using bootstrapping methods (e.g., [35, 36]).

To cope with the problems of "wrapper generation" and "wrapper maintenance" rule-based methods have been especially popular in recent years. Some techniques for generating rules in the realm of text extraction are called *"wrapper induction"* methods. These techniques have proved to be rather successful for IE tasks in their intended domains, which are collections of documents such as web pages generated from a template script [37, 38, 39]. However, wrapper induction methods do only extend well to documents specific to the induced rules.

In *semi-automatic* wrapper generation Machine Learning approaches are applied. Tools may support the design of the wrapper. Some approaches offer a declarative interface where the user shows the system what information to extract (e.g., [38, 39]). *Automatic* wrapper generation tools use unsupervised learning techniques. Therefore, no training sets are necessary, just a post-generation tuning (e.g., [40, 41]).

When developing an IE system one has to incorporate numerous criteria to decide which approach to apply [23]. These are the availability of training data, which counts for an automatic learning approach, or the availability of linguistic resources and knowledge engineers, where the Knowledge Engineering approach may be favored. Also the level of performance required and the stability of the final specifications are important factors which may be better fostered by the Knowledge Engineering approach.

Required extraction rules

Rules for IE are developed from a training set of documents - in our case clinical practice guidelines. For our system, we defined patterns on three levels, whereas patterns at a certain level serve as concept classes in the preceding levels: (1) phrase level patterns, (2) sentence level patterns, and (3) discourse level patterns. Pattern rules were designed using the atomic approach [23]. Thereby, a domain module is built that recognizes the arguments to an event and combines them into template structures strictly on the basis of intelligent guesses rather than syntactic relationships. In doing so domain-relevant events are assumed for any recognized entities, leading to high recall, but much overgeneration, and thus low precision. Further development would result improving filters and heuristics for combining the atomic elements, improving precision.

Medical terms (i.e., drug agents, surgical procedures, and diagnostic terms) are based on a subset of the Medical Subject Headings (MeSH)[11] of the United States National Library of Medicine. We adapted them according for missing terms, different wordings, acronyms, and varying categorization.

Phrase level patterns. They are used for identifying basic entities, such as *time*, *dosage*, *iteration*, and *condition* expressions, which build the attributes of actions. They are defined by regular expressions.

Sentence level patterns. They use phrase level patterns, medical terms, and trigger words for the medical terms to identify medical actions and their attributes. The trigger words are mainly verbs and indicate the application of a therapy (e.g., the administration of a drug agent or the implementation of a surgical procedure) or the avoidance of a therapy. Sentence level patterns are delimiter-based and use syntactic constraints. We can categorize the patterns in two groups: (1) patterns for free text and (2) patterns for telegraphic text.

The former are applied to free text, which has a grammatical structure and is usually identified in paragraphs, but also in list elements. These patterns indicate that therapy instruments (i.e., agent terms and surgical procedures) combined with trigger terms (e.g., 'activate', 'indicate', 'perform', 'prescribe') appearing in the same clause identify relevant sentences. The particular clauses must not be condition clauses. Phrase level patterns, such as <`dosage`>, <`duration`>, <`condition`>, and so on can be arbitrarily combined

[11] http://www.nlm.nih.gov/mesh/ (Accessed: Dec. 12, 2004)

with <therapy instrument> <trigger> pairs. But information concerning a treatment recommendation can be distributed in several sentences. These sentences including additional information (e.g., *'The standard dose is 40 to 45 mg/kg/day.'*) neither contain a therapy instrument nor a trigger term, but also have to be identified by sentence patterns.

Telegraphic text patterns are applicable in list elements. In these elements often ungrammatical text is formulated. Often, only a therapy instrument indicates the relevancy of an element. Other patterns exist for list elements indicating that these elements are relevant if within their context or in the paragraph preceding the list special terms appear. These terms (i.e., 'remedy', 'remedies', 'measure', 'measures', 'medication', 'medications') are important, because they specify actions that may not contain therapy instruments in the form of agent terms or surgical procedures (e.g., *'Maintain adequate hydration (drink 6 to 10 glasses of liquid a day to thin mucus)'*).

Discourse level patterns. They are based on *sentence level patterns*, but are augmented to consider the structure and the layout of the documents. They are used to categorize sentences, merge them to actions, and find relationships between actions to structure them. To accomplish the latter task we analyzed treatment processes contained in the guidelines and detected the following processes, whereas some of them are identified by discourse level patterns:

- Processes without temporal dependencies
- Sequential processes
- Processes containing subprocesses
- Selections of processes
- Recurring processes

Gaining Process Information

To extract processes from CPGs we proceed in several steps which serve as filter segments of text containing treatment instructions from the documents and to generate processes. We propose a two-step approach to gain a representation that is independent of the subsequent guideline representation language (cp. Figure 9). The first step is to extract relevant sentences containing treatment recommendations by marking-up the original guideline document. The subsequent step is to combine several sentences to one action and to structure the actions and detect relations among them. These two steps provide a basis for the subsequent transformation of the process information into any guideline representation language.

Extracting relevant sentences

This task is a first step towards our final guideline representation. We will achieve it by two modules: (1) the segmentation & filtering module and (2) the template generation module (see Figure 10 for an overview).

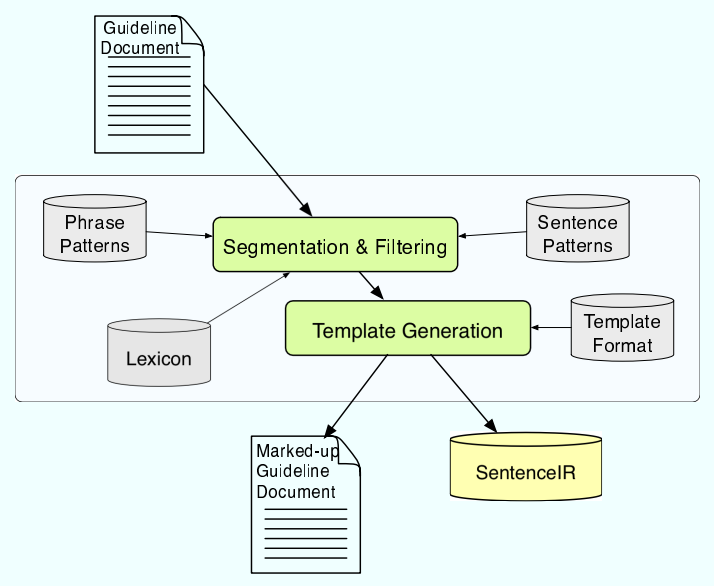


Fig. 10. Detecting relevant sentences. We split this task into two modules: (1) segmentation & filtering module and (2) template generation module.

This first intermediate step is especially important as not the entire content of a guideline contains processes, which are to be modeled. Although health care consists of the three stages observation, diagnosis, and therapy [42] we only want to model the control flow regarding the therapy. Only about 20 % of sentences of a guideline are of interest for modeling processes. On this account it is important to select the relevant sentences for modeling.

Thus, this task performs an automatic mark-up of sentences that are utilized to process the subsequent steps.

Segmentation & filtering. Detecting relevant sentences is a challenging task, which we undertake in two steps: (1) detecting irrelevant text parts to exclude them from further processing and (2) detecting relevant sentences. Irrelevant text parts (i.e., sections, paragraphs) are associated with diagnosis, symptoms, or etiology, relevant sentences describe actions of a treatment process.

The first filtering occurs at the section level. Sections in the document with captions indicating diagnosis or symptom declarations will be omitted in further processing. We can identify these captions by keywords such as 'history', 'diagnosis', 'symptom', 'clinical assessment', 'risk factor', and so on.

Detecting relevant sentences is not a trivial task. First, we parse the entire document and split it into sentences. Then we process every sentence with regard to its context within the document and its group affiliation. Thereby, the context is obtained by captions (e.g., *'Acute Pharyngitis Algorithm Annota-*

tions | Treatment | Recommendations:') and a group contains sentences from the same paragraph or the same list, if there are no sublists. Each sentence is then checked for relevance by applying *sentence level patterns*.

Template generation. After having collected the relevant sentences from the guideline, we can proceed with generating the IR *SentenceIR*. We generate two files: one file listing all relevant sentences and the marked-up guideline document (Listing 1 shows a part of a marked-up guideline document). Both are linked by applying the same id to the same sentences in order to give the user the possibility to see the *SentenceIR* representation and its corresponding textual representation using the DELT/A tool (cp. Figure 11). The presentation of the template file and the guideline document are as simple as possible in order to support the user by detecting all relevant sentences.

Listing 1. Excerpt of a source listing of the marked-up guideline document "Evidence-based clinical practice guideline for children with acute bacterial sinusitis in children 1 to 18 years of age" [43]. Relevant sentences are enclosed by HTML-like "a" tags.

```
<li>
   <a  id="delta:8">In children with risk factors for Streptococcus
        pneumoniae, it is recommended that Amoxicillin, high dose (80 to
         90 mg/kg/day) or Augmentin (with high dose amoxicillin
        component) be utilized as first-line therapy.
   </a>
   <ul  type="disc">
      <li>
         <a  id="delta:9">Note: Failure with amoxicillin is likely to
              be due to resistant Streptococcus pneumoniae, Haemophilus
              influenzae, or Moraxella catarrhalis.
         </a>
         <a  id="delta:10">High dose amoxicillin will overcome
              Streptococcus pneumoniae resistance (changes in
              penicillin-binding proteins).
         </a>
         The clavulanic acid component of Augmentin is active
         against resistant Haemophilus influenzae and Moraxella
         catarrhalis (B-lactamase enzyme).
      </li>
   </ul>
</li>
```

Extracting required information and finding processes

The information contained in *SentenceIR* and the marked-up guideline document are the input for the next task (see Figure 12 for an overview). Its goal is to structure relevant sentences and find relationships between sentences. Again, the output of this task should be represented in a format that is independent of any desired guideline representation format.

Structure extraction. In this task we obtain the context of each sentence by means of hierarchical groups which is necessary for other subtasks, especially the merging and grouping and the process extraction. Every action is assigned to one group. The context of a sentence defines the affiliation to a group and

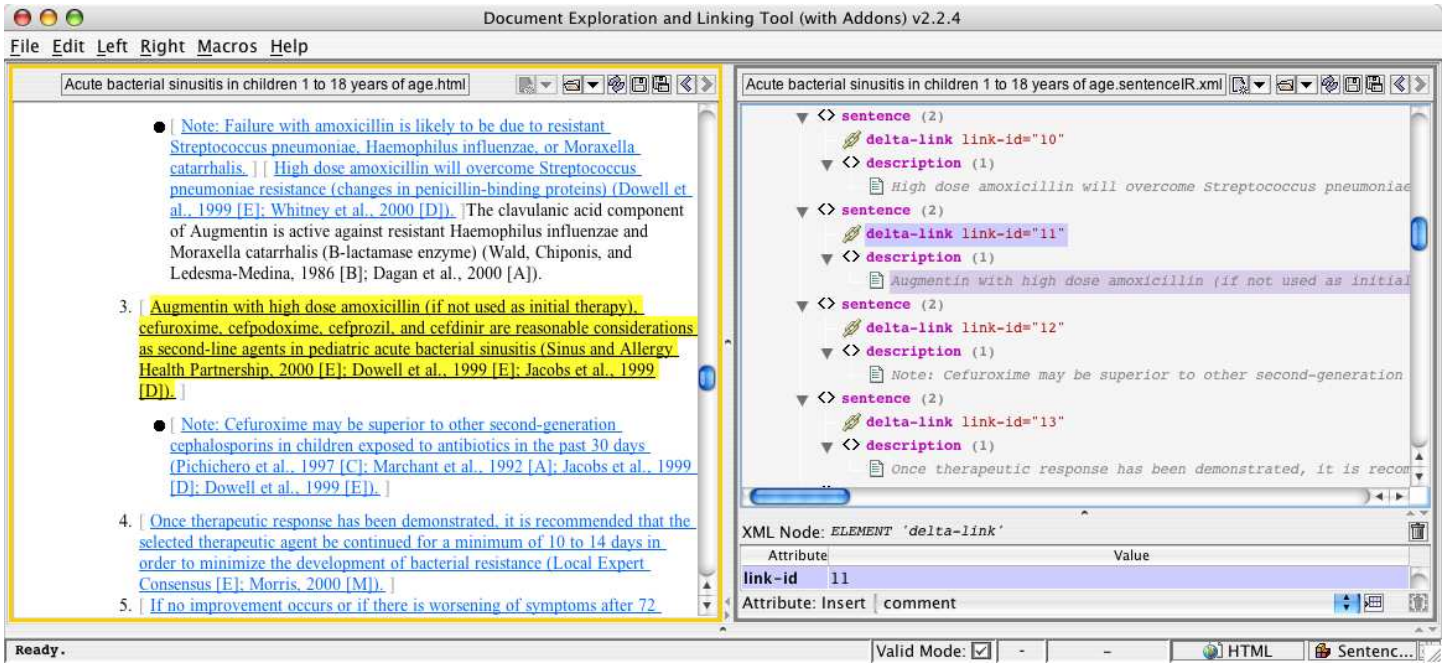


Fig. 11. The DELT/A tool showing the marked-up guideline on the left side and the *SentenceIR* file containing a list of all extracted sentences on the right side.

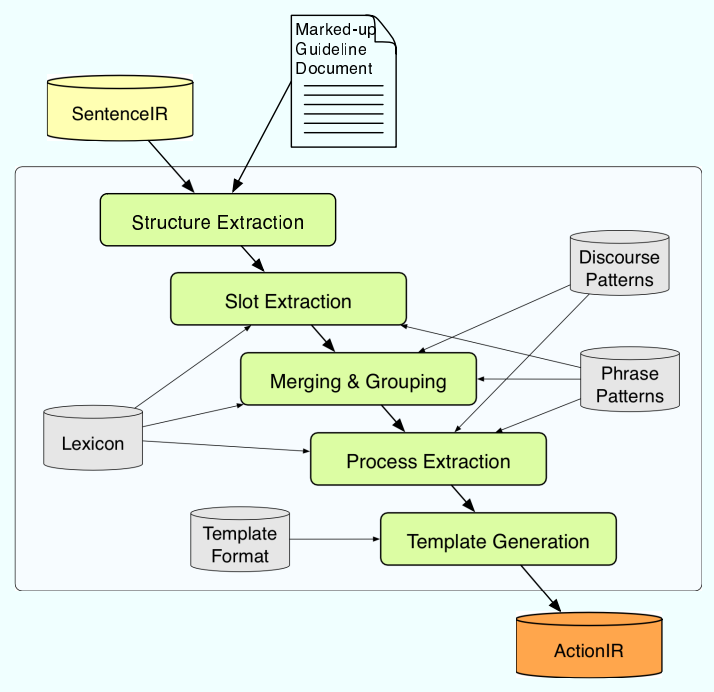


Fig. 12. Finding processes and extracting required information. We split this task into five modules (i.e., the structure extraction module, the slot extraction module, the merging & grouping module, the process extraction module, and the template generation module).

is defined by the sentence's position in the hierarchical structure. We use the superior headings that establish several context items.

Slot extraction. This module is used to extract therapy instruments (i.e., agent terms and surgical procedures), dosage information in case of a drug administration, the duration of the therapy action, the iteration information of the action, as well as conditions which have to be fulfilled to perform an action. It uses both the lexicon and the *phrase level patterns*.

Merging & grouping. In this module we categorize sentences in actions or negative actions and annotations. Annotations always belong to at least one action (or negative action). They cannot exist alone. This module extensively applies *discourse level patterns*.

First, we check whether a sentence describes an action or a negative action. Negative actions are instructions that an action should not be performed, often under specific conditions (e.g., *'Do not use aspirin with children and teenagers because it may increase the risk of Reyes syndrome.'*). Most guideline representation languages will handle such actions by inverting the condition.

Languages may exist which will handle these in other ways. Therefore, we provide a representation for such actions that can be used in a general way.

Furthermore, we identify annotations and assign them to their corresponding actions or negative actions using *name-alias coreferencing* and *definite description coreferencing* based on therapy instruments and their hypernyms. We do not apply pronoun-antecedent coreference.

Process extraction. To group actions and to detect relationships between actions we use *discourse level patterns*. We will describe those used by this module below.

The default relationship among processes is that there is no synchronization in their execution. To group actions to a *selection* they must fulfill the following requirements: (1) the actions have to belong to the same group, and (2) agents or surgical procedures must have the same superordinate. For instance, processes describing the administration of *Erythromycin*, *Cephalexin*, and *Clindamycin* within one group are combined in a *selection*, as all these agents are antibiotics. If actions are grouped in a selection, one of these actions has to be selected to be executed.

Furthermore, we try to detect relations between actions that are explicitly mentioned within the text as well as relations that are implicitly given by the document structure. The former is very difficult to detect, as we often cannot detect the reference of the relation within the text (e.g., *'After 10 to 14 days of failure of first line antibiotic ...'*). Nevertheless, we found heuristics that arrange actions or action groups if the reference is unambiguously extractable out of the text. These heuristics can be grouped in two categories: (1) detecting sentences describing relations between actions, and (2) detecting actions that are described in the preceding heuristic. A relation is mainly identifiable by a relation term (e.g., 'before', 'after', 'during', 'while'). If such a term appears, we are searching for therapy instruments, as these describe most of our actions. After we have detected these terms, we search for actions containing the particular instruments. If we have found both the source action and the destination action we can create a new relation.

We use patterns of the document structure (e.g., *'Further Treatment'* appears `after` *'Treatment'* or *'Treatment'* appears `before` *'Follow-Up'*) to detect implicitly given relations. These patterns are part of discourse level patterns to determine relations between several groups.

Template generation. The template of this IR has to contain actions as well as their relations. It has to be simple and concise and it has to illustrate from which original data the current information was built. We split the new *ActionIR* template in three parts: (1) an area for actions, (2) an area for relations, and (3) an area for the structure illustrating the hierarchy and nesting of groups.

An *action* contains the action sentence, the assigned annotation sentences, treatment instruments and their MeSH ids, information about the dosage, duration, or iteration of a drug administration, and conditions. If the action is

part of a selection, it is stated by the selection id. DELT/A links are inherited from the *SentenceIR* representation in order to provide the traceability of the process from both the original guideline document and the *SentenceIR* document. Listing 2 shows an example instance.

Relations are stated by their type (e.g., succeeding, preceding, overlapping) and the concerned actions by their DELT/A ids.

Listing 2. Action instance of an *ActionIR* template for the guideline "Evidence-based clinical practice guideline for children with acute bacterial sinusitis in children 1 to 18 years of age" [43].

```
<action id="8" parent="5" group="18" selection="0">
   <delta-link link-id="8"/>
   <description>In the child with no risk factors for
        penicillin-resistant Streptococcus pneumoniae standard dose
        amoxicillin or Augmentin (with standard dose Amoxicillin
        component) may be considered as initial therapy.
   </description>
   <agents>
      <agent MeSH="D000658" name="amoxicillin"/>
      <agent MeSH="D019980" name="Augmentin"/>
   </agents>
   <condition>
      <item>In the child with no risk factors for penicillin-resistant
           Streptococcus pneumoniae
      </item>
   </condition>
   <annotations>
      <annotation>Note: Forty-six percent of isolates at Children's
           Hospital Medical Center of Cincinnati, Ohio have intermediate
            or high Penicillin-resistant Streptococcus pneumoniae and
           local data supports that 15% of children locally may fail
           initial therapy with standard dose amoxicillin.
         <delta-link link-id="9"/>
      </annotation>
   </annotations>
   <context>
      <item>Antibiotic Treatment</item>
   </context>
</action>
```

Modeling Plans in *Asbru*

Now we wanted to verify whether the information obtained by the process extraction task is in a format that can be utilized to transform it into a computer-interpretable guideline representation. We have chosen the language Asbru [3] which was developed to embody CPGs as time-oriented skeletal plans. We process the representation *AsbruIR* to subsequently generate an *Asbru* guideline.

A step towards Asbru

In the next step we extract Asbru-specific information and integrate it in an other IR. This new format contains several actions and their temporal specifications to generate both atomic and cyclical plans in Asbru. The IR *AsbruIR* contains the following refined Asbru-specific data:

- Temporal information of durations and iterations are refined and calculated if necessary
- Conditions are refined and classified in categories (e.g., 'diagnosis', 'patient', 'allergy') if possible
- Interval relations are modeled
- Actions and structure are merged
- Auxiliary actions are inserted

Figure 13 gives an overview of the steps required to obtain the new representation.

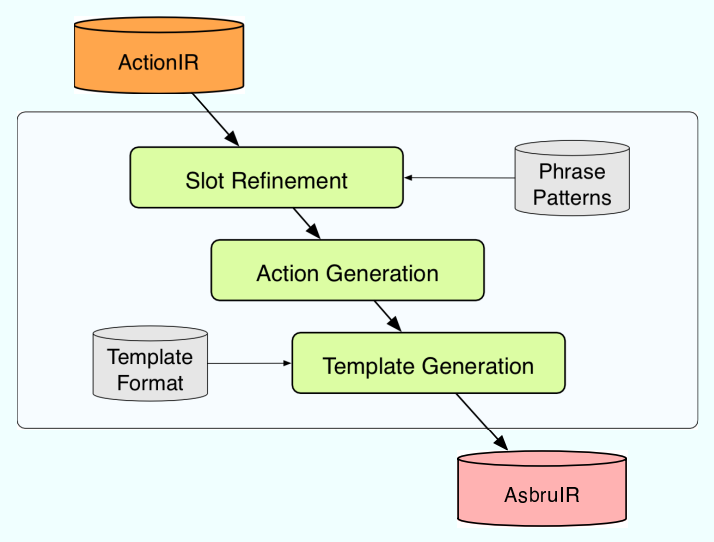


Fig. 13. Refining process information and generating auxiliary actions. We split this task into three modules (i.e., slot refinement module, action generation module, and template generation module).

Slot Refinement. A lot of the information we have extracted so far is not in an Asbru-interpretable format. For instance, phrases, such as *every 4 to 6 weeks*, describe an iteration, but to use it in Asbru it has to be itemized in values and units. Therefore, we use phrase patterns with an Asbru-specific emphasis. In case of an iteration we also define its type (e.g., iterations defined by the frequency or by the period between two recurrences). Similarly we proceed with 'duration' instances.

To better cope with various conditions we classify them into patient-, disease-, and allergy-specific conditions. Furthermore, in case of a negative action we mark each condition.

Interval relations are integrated in the particular actions by using temporal conditions. Asbru is able to cope with both incomplete and uncertain temporal

information and provides a powerful means to represent intervals: the *time annotation* (see Figure 14). [44] shows in detail which possibilities are available to model such interval relations.

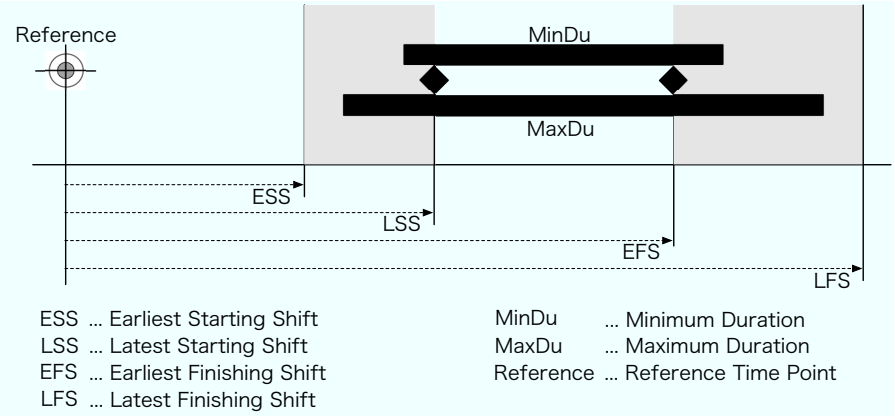


Fig. 14. Time interval in Asbru. The grey areas indicate periods when the action has to start and accordingly finish.

Action Generation. This module addresses the merging of the action section and the structure section as well as the generation of auxiliary actions.

The *structure* section of the *ActionIR* representation contains the hierarchy of the guideline document and is now filled with the particular actions.

For *select-one-of* actions we have to generate a special action which controls the choice of the particular actions. Although this kind of plan often appears in guidelines Asbru does not provide a separate modeling concept. Thus, the modeling is elaborate and we will support it in this IR. *Select-one-of* actions must have a parent action able to invoke each single action including a constraint that the parent action has to complete after the finishing of the selected action. Listing 3 shows an example.

Listing 3. Example of a parent action constituting a *select-one-of* relation for actions with plan ids '6' and '7'.

```
<action id="SELECT_0" parent="5" group="16">
   <condition>
      <item type="complete-condition" plan="6" state="completed"/>
      <item type="complete-condition" plan="7" state="completed"/>
   </condition>
</action>
```

Further auxiliary actions are generated in order to represent the entire guideline hierarchy.

Template Generation. For this representation we focus on automatic processing to generate an Asbru protocol. For the human user it is difficult to get

an overall outline of the guideline as actions are displayed in their hierarchical structure. Due to the refined itemization the information is not so easily readable (in the sense of understandable). Therefore, we additionally state it as string information. Listing 4 shows an example of an *AsbruIR* instance.

Listing 4. Instance of an *AsbruIR* template.

```
<action id="25" parent="SELECT_1" group="18" selection="1">
   <delta-link link-id="25" />
   <description>For those allergic to amoxicillin:
        Trimethoprim-sulfamethoxazole (TMP/SMX): one double strength tab
         BID 10 days
   </description>
   <agents>
      <agent MeSH="D015662" name="Trimethoprim-sulfamethoxazole">
         <iteration term="BID" specification="CYCLICAL">
            <frequency value="12" unit="h" />
            <minimum value="2"  unit="h" />
            <maximum value="10" unit="h" />
         </iteration>
         <duration term="10 days">
            <minimum value="10" unit="d" />
            <maximum value="10" unit="d" />
         </duration>
      </agent>
   </agents>
   <condition>
      <item allergy="amoxicillin">For those allergic to amoxicillin</
          item>
   </condition>
   <annotations>
      <annotation>Trimethoprim-sulfamethoxazole (TMP/SMX) is a
          potential first-line antibiotic.
         <delta-link link-id="28" />
      </annotation>
      <annotation>Studies have shown effectiveness with 3 to 14 days.
         <delta-link link-id="32" />
      </annotation>
   </annotations>
</action>
```

Obtaining Asbru

The final step in our approach is the generation of an Asbru guideline. We accomplish this task using XSLT[12].

We therefore built several XSLT templates to transform the required Asbru segments. If these segments are combined they might represent an Asbru guideline. Starting from the IR *AsbruIR* the templates have to map the following concepts: plans and parameter definitions. For more information about the Asbru language and its modeling requirements for processes refer to [44].

Plans. They are the basic building blocks of Asbru guidelines. Our XSLT templates have to constitute several kinds of plans which arise from *AsbruIR*.

- *Atomic plans.* This plan cannot be refined. Example: user-performed plans.

[12] XSLT is an acronym for eXtensible Stylesheet Language (XSL) Transformation. It is an XML-based language used for the transformation of XML documents.

- *Cyclical plans.* This plan iteratively calls another plan. The modeling depends on whether the iteration is frequency-based or period-based.
- *Plans containing subplans.* Plans can be nested. Due to the hierarchical structure of guidelines this is a very frequent concept. Asbru provides the possibility to model various kinds of synchronization among the subplans.

Plans contain several optional child elements. For our modeling we only use 'conditions' and 'plan-body'.

- *Conditions.* They control the execution of a plan by enabling plan instances to receive a certain plan state. Figure 15 describes the conditions that must apply to switch from one state to an other.
  Conditions can only map temporal constraints. The concept of 'conditions' as appearing in *ActionIR* is not equal to Asbru's 'conditions' concept and thus has to be modeled by other concepts which appear in the 'plan-body' element.

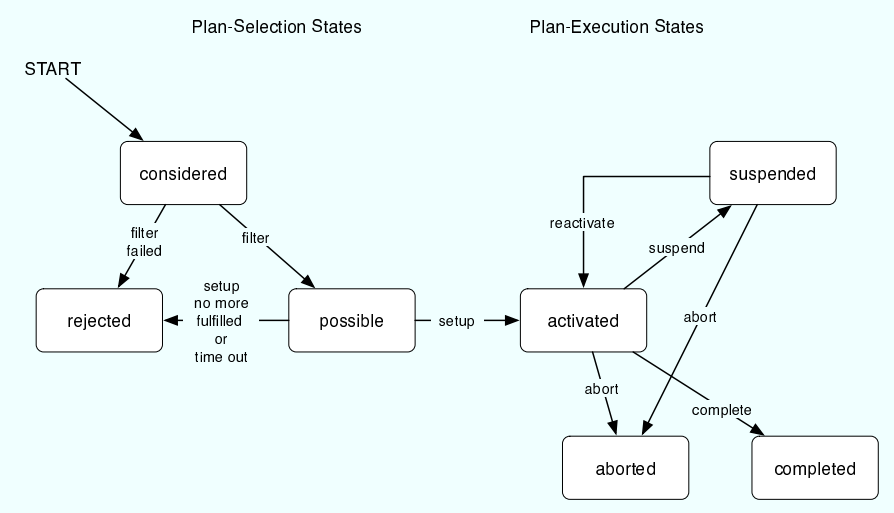


Fig. 15. Plan states and conditions in Asbru. The arrows specify the type of condition that must apply to switch from one state to another.

- *Plan-body.* It contains the information about the events when executing the plan. There are several possibilities, which are all described in [45]. For our modeling we only use
  - Subplans. A set of plan steps performed in parallel or sequentially.
  - Cyclical-plan. A plan repeated several times.
  - Single-step. A single step of plan execution; for instance, a plan activation.
  - User-performed. This plan is executed through some action by the user, which is not further modeled in the system.

Parameter definitions. In Asbru variables are referred to as parameters. Parameters may have quantitative as well as qualitative character and are defined in the 'domain-defs' section. In many cases the definition of parameters has to be done manually as they often reflect implicit or tacit knowledge.

We use parameters in many condition statements. Listing 5 shows an example of parameter definitions for a guideline.

Listing 5. Parameter definitions for an Asbru guideline.

```
<domain-defs>
   <domain name="Acute_pharyngitis">
      <parameter-group title="Diagnosis parameters">
         <parameter-def name="for_treatment_of_culture_
              positive_cases_of_GABS_pharyngitis" required="no" type="
              any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Diagnosis is GABS pharyngitis?"/>
         </parameter-def>
         <parameter-def name="for_streptococcal_pharyngitis" required="
              no" type="any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Diagnosis is streptococcal pharyngitis?"/>
         </parameter-def>
         <parameter-def name="for_strep_pharyngitis" required="no" type
              ="any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Diagnosis is strep pharyngitis?"/>
         </parameter-def>
         <parameter-def name="for_treatment_of_ GABS_pharyngitis"
              required="no" type="any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Diagnosis is GABS pharyngitis?"/>
         </parameter-def>
      </parameter-group>
      <parameter-group title="Allergy parameters">
         <parameter-def name="In_PCN-allergic_patients" required="no"
              type="any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Patient is allergic to PCN?"/>
         </parameter-def>
         <parameter-def name="In_PCN-_and_erythromycin-
              allergic_patients" required="no" type="any">
            <raw-data-def mode="manual" use-as-context="no" user-text="
                Patient is allergic to PCN and erythromycin?"/>
         </parameter-def>
      </parameter-group>
   </domain>
</domain-defs>
```

Using different templates we are able to generate Asbru plans with various levels of detail. For instance, we can create plans at a very low level where an action in an *AsbruIR* representation corresponds to exactly one Asbru plan – no matter if the action contains iterative instructions or a bundle of treatment instructions. The highest level in Asbru modeling would be the itemization of iterative instructions and further subplans.

AsbruIR is a mixture between single- and multi-slot templates. It consists of actions, whereas one action might consist of several sentences (i.e. action sentence and several annotation sentences). But this action may consist of several (alternative) sub-actions, which is shown in the 'agents' section of the

template: each agent is separately presented and can therefore form a separate plan in Asbru.

Furthermore, non-temporal conditions (e.g., *In case of penicillin allergy: oral cephalexin 750 mg x 2 or cefadroxil 1 g x 1.*) have to be modeled by 'if-then-else' statements containing subplans that are activated depending on the condition's result (see Listing 6).

Listing 6. Asbru: Plan for an `if-then-else` statement. The plan contains a subplan including an `ask` statement, where the parameter value is queried, and an `if-then-else` statement, where the value is compared to a constant. In case of match the `then-branch` gets executed.

```
<plan name="PLAN_7" title="In case of penicillin allergy: oral
    cephalexin 750 mg x 2 or cefadroxil 1 g x 1 (Deeter et al., 1992;
    DARE-953519, 1999) [A].">
  <delta-link link-id="7"/>
  <plan-body>
    <subplans retry-aborted-subplans="no" type="any-order"
        wait-for-optional-subplans="no">
      <wait-for>
        <all/>
      </wait-for>
      <ask>
        <parameter-ref name="In_case_of_penicillin_allergy"/>
        <time-out>
          <now/>
        </time-out>
      </ask>
      <if-then-else>
        <simple-condition>
          <comparison type="equal">
            <left-hand-side>
              <parameter-ref name="In_case_of_penicillin_allergy
                  "/>
            </left-hand-side>
            <right-hand-side>
              <qualitative-constant value="yes"/>
            </right-hand-side>
          </comparison>
        </simple-condition>
        <then-branch>
          <plan-activation>
            <plan-schema name="SELECT_PLAN_7">
              <delta-link link-id="7"/>
            </plan-schema>
          </plan-activation>
        </then-branch>
      </if-then-else>
    </subplans>
  </plan-body>
</plan>
```

Modeling Asbru is very tedious. But based on well structured and extensive data it is possible to create the necessary Asbru statements using sophisticated XSLT templates. Due to their size and complex syntax Asbru guidelines are not comprehensible for a human user, but they can be executed in a computer-supported way.

3.3 Results

In order to evaluate our methodology we applied a framework for translating otolaryngology guidelines into Asbru. We therefore chose evidence-based guidelines for treatment and management featuring temporal aspects of flows and divided the set of documents into a training set of six guidelines and a test set of twelve guidelines. The training set was used to adapt the extraction rules according to special characteristics of the clinical specialty. The test set was then used to evaluate both the IE part and our proposed methodology. Two persons participating the evaluation generated key target templates for the tasks for each guideline which were compared to the automatically generated output templates of LASSIE. The input data for the second and every subsequent step were the key target templates of the particular previous step. Based on this data we compiled the number of total possible correct responses (POS), the number or correct values (COR), and the number of values generated by the system (ACT). Using these values we generated the recall and precision scores, which are used to measure IE systems [46].

$$recall = \frac{COR}{POS} \tag{1}$$

$$precision = \frac{COR}{ACT} \tag{2}$$

Recall measures the ratio of correct information extracted from the texts against all the available information present in the text. That means, it specifies how well the system finds what you want. Precision measures the ratio of correct information that was extracted against all the information that was extracted. That means, it specifies how well the system filters what you do not want.

It is hardly possible to develop a system which delivers only correct results. Systems are often optimized with respect to one score.

Thus, we have to consider two cases for the *mark-up task*:

1. The system should detect almost all relevant sentences and probably will spuriously detect some irrelevant sentences, too.
2. The system should hardly detect irrelevant sentences as relevant and probably will ignore some relevant sentences.

Systems with an emphasis on the first case will gain a higher recall, but may concurrently derive a decreased precision. Systems with an emphasis on the second case will gain a higher precision at the expense of a decreased recall. To optimize the benefit of this task it is more important to provide almost all relevant sentences rather than reading the remaining ones, about 80 %, of the guideline to detect the lacking relevant sentences. After continuous improvement we gained a recall score of 90.8 % and a precision score of 94.9 % which constitutes a benefit even if the recall score is less than 100 %.

For the *process extraction task* overall recall and precision are 84.0 % and 86.8 %, respectively. Apart from the evaluation of the overall task we also analyzed the *slot extraction* and *sentence categorization and assignment* subtasks. Thereby, the optimization regarding recall and precision has to be done individually for each of the subtasks and slots, respectively. For detailed evaluation results see [44].

Furthermore, we also wanted to constitute the benefit of the process steps by the users. We measured the benefit on the basis of the effort manually modeling of adjusting the processes using the DELT/A tool. In [44] we have shown that users benefit from automating the last two transformation steps (i.e., obtaining the *AsbruIR* and *Asbru* representations) and that based on the correct representation in *ActionIR* an almost errorless transformation into *Asbru* is possible.

4 Conclusion

Modeling CPGs is a complex task which has to be assisted by both physicians and knowledge engineers. Bearing those two user groups in mind a method is demanded supporting them in their particular fields of functions: the physicians have to be less overcharged by the formal specifications and the knowledge engineers have to be fostered by providing medical knowledge. Based on this conceptual formulation and already existing methods and tools we developed a stepwise procedure for modeling treatment processes using IE – the LASSIE methodology.

Findings in our evaluation discussed in [44] indicate that using semi-automatic, step-wise IE methods are a valuable instrument to formalize CPGs. We have developed several IE and transformation rules, which we integrated in a framework and applied them to several guidelines of the specialty of otolaryngology. Thereby, we firstly generate a simple representation of treatment instructions (i.e., actions), which are independent from the final guideline representation language. Based from this independent representation we can secondly transform the information in further steps into the guideline representation languages. To proof our methodology we applied the framework to formalize guidelines in the formal *Asbru* plan representation.

Nevertheless, some problems and shortcomings of guideline modeling with LASSIE are not solved so far. Although the review of the modeling process using DELT/A is a great support, its representation and usage is unfamiliar for physicians. They have difficulties using DELT/A as most of them are not familiar with XML, the concept of macros, and any representation format that is not pure natural language text.

Anyhow, LASSIE offers distinct benefits, in particular:

- Automating of the modeling process
- Disburdening of the physicians in the modeling process by providing a medical knowledge base

- Structuring of the guideline information
- Decomposition of guidelines into parts containing various kinds of information
- Making the modeling process traceable and comprehensible
- The applicability for many guideline representation languages
- Supporting the guideline development process in order to better structure guidelines, identifying ambiguities, inconsistencies, and incompleteness

Furthermore, the methodology may also influence the application of the concept of *'living guidelines'*, an approach to update guidelines on a more continuous basis than the usual practice of revision every two to five years. Scientific and pragmatic knowledge is growing faster every year and therefore a guideline is a static document, which cannot be modified easily. To become flexible, adaptable documents the aim is to develop guidelines, which present up-to-date and state-of-the-art knowledge to practitioners. To make this possible, guidelines have to be modular in structure, so that only part of a guideline must be adjusted and not the whole document needs revision.

References

1. Lina T. Kohn, Janet M. Corrigan, and Molla S. Donaldson, editors. *To Err Is Human: Building a Safer Health System.* National Academy Press, Washington, D.C., 2000.
2. Marilyn J. Field and Kathleen N. Lohr, editors. *Clinical Practice Guidelines: Directions for a New Program.* National Academies Press, Institute of Medicine, Washington DC, 1990.
3. Paul A. de Clercq, Johannes A. Blom, Hendrikus H. M. Korsten, and Arie Hasman. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial Intelligence in Medicine*, 31(1):1–27, May 2004.
4. Mor Peleg, Samson W. Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A. Greenes, Richard Hall, Peter D. Johnson, Neill Jones, Anand Kumar, Silvia Miksch, Silvana Quaglini, Andreas Seyfang, Edward H. Shortliffe, and Mario Stefanelli. Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. *Journal of the American Medical Informatics Association (JAMIA)*, 10(1):52–68, Jan-Feb 2003.
5. Voitech Svátek and Marek Růžička. Step-by-step mark-up of medical guideline documents. *International Journal of Medical Informatics*, 70(2-3):319–335, July 2003.
6. Kristi-Anne Polvani, Abha Agrawal, Bryant Karras, Aniruddha Deshpande, and Richard Shiffman. *GEM Cutter Manual.* Yale Center for Medical Informatics, 2000.
7. Richard N. Shiffman, Bryant 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 Association (JAMIA)*, 7(5):488–498, 2000.
8. Gem cutter: Screenshot. http://gem.med.yale.edu/GEM_Cutter/gem_cutter.htm, 2005. [retrieved: Oct. 20, 2005].

9. Peter Votruba, Silvia Miksch, and Robert Kosara. Facilitating knowledge maintenance of clinical guidelines and protocols. In Marius Fieschi, Enrico Coiera, and Yu-Chuan Jack Li, editors, *Proceedings from the Medinfo 2004 World Congress on Medical Informatics*, pages 57–61. IOS Press, 2004.
10. Yuval Shahar, Ohad Young, Erez Shalom, Alon Mayaffit, Robert Moskovitch, Alon Hessing, and Maya Galperin. DEGEL: A hybrid, multiple-ontology framework for specification and retrieval of clinical guidelines. In Michel Dojat, Elpida Keravnou, and Pedro Barahona, editors, *Proceedings of the 9th Conference on Artificial Intelligence in Medicine in Europe, AIME 2003*, volume 2780 of *LNAI*, pages 122–131, Protaras, Cyprus, 2003. Springer Verlag.
11. Robert Kosara and Silvia Miksch. Metaphors of Movement: A Visualization and User Interface for Time-Oriented, Skeletal Plans. *Artificial Intelligence in Medicine, Special Issue: Information Visualization in Medicine*, 22(2):111–131, May 2001.
12. John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The Evolution of Protégé: An Environment for Knowledge-based Systems Development. *International Journal of Human Computer Studies*, 58(1):89–123, 2003.
13. David R. Sutton and John Fox. The syntax and semantics of the PRO*forma* guideline modeling language. *Journal of the American Medical Informatics Association (JAMIA)*, 10(5):433–443, Sep / Oct 2003.
14. Arezzo overview. http://www.infermed.com/arezzo/arezzo-components/, 2006. [retrieved: March 22, 2006].
15. Arezzo composer. http://www.infermed.com/arezzo/arezzo-create-guideline/, 2006. [retrieved: March 22, 2006].
16. R. Steele and John Fox. Tallis PROforma Primer – Introduction to PROforma Language and Software with Worked Examples. Technical report, Advanced Computation Laboratory, Cancer Research, London, UK, 2002.
17. M. Balser, O. Coltell, J. van Croonenborg, C. Duelli, F. van Harmelen, A. Jovell, P. Lucas, M. Marcos, S. Miksch, W. Reif, K. Rosenbrand, A. Seyfang, and A. ten Teije. Protocure: Integrating formal methods in the development process of medical guidelines and protocols. In Katharina Kaiser, Silvia Miksch, and Samson W. Tu, editors, *Computer-based Support for Clinical Guidelines and Protocols. Proceedings of the Symposium on Computerized Guidelines and Protocols (CGP 2004)*, volume 101 of *Studies in Health Technology and Informatics*, pages 103–107, Prague, Czech Republic, 2004. IOS Press.
18. Andreas Seyfang, Silvia Miksch, Cristina Polo-Conde, Jolanda Wittenberg, Mar Marcos, and Kitty Rosenbrand. MHB - a many-headed bridge between informal and formal guideline representations. In *Proceedings of the 10th Conference on Artificial Intelligence in Medicine in Europe, AIME 2005*, volume 3581 of *LNAI*, pages 146–150, Aberdeen, UK, 2005. Springer Verlag.
19. James R. Campbell, Samson W. Tu, James G. Mansfield, Julie I. Boyer, James McClay, Craig Parker, Prabhu Ram, Sidna M. Scheitel, and Kevin McDonald. The SAGE guideline model: A knowledge representation framework for encoding interoperable clinical practice guidelines. In *Proceedings of the AMIA Annual Symposium*, November 2003.
20. Andreas Seyfang, Silvia Miksch, Mar Marcos, Jolanda Wittenberg, Cristina Polo-Conde, and Kitty Rosenbrand. Bridging the gap between informatl and formal guideline representations. In *European Conference on Artificial Intelligence (ECAI-2006)*, 2006, forthcoming.

21. Samson W. Tu, Mark A. Musen, Ravi Shankar, James Campbell, Karen Hrabak, James McClay, Stanley M. Huff, Robert McClure, Craig Parker, Roberto Rocha, Robert Abarbanel, Nick Beard, Julie Glasgow, Guy Mansfield, Prabhu Ram, Qin Ye, Eric Mays, Tony Weida, Christopher G. Chute, Kevin McDonald, David Mohr, Mark A. Nyman, Sidna Scheital, Harold Solbrig, David A. Zill, and Mary K. Goldstein. Modeling guidelines for integration into clinical workflow. In Marius Fieschi, Enrico Coiera, and Yu-Chuan Jack Li, editors, *Proceedings from the Medinfo 2004 World Congress on Medical Informatics*, pages 174–178. IOS Press, 2004.
22. Katharina Kaiser, Cem Akkaya, and Silvia Miksch. How can information extraction ease formalizing treatment processes in clinical practice guidelines? A method and its evaluation. *Artificial Intelligence in Medicine*, 2006. to appear.
23. Douglas E. Appelt. Introduction to information extraction. *AI Communications*, 12:161–172, 1999.
24. Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, January 1996.
25. Roman Yangarber. *Scenario Customization for Information Extraction*. PhD thesis, New York University, New York, January 2001.
26. Nick Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper Induction for Information Extraction. In *International Joint Conference on Artificial Intelligence*, Nagoya, 1997.
27. Jerry R. Hobbs, Douglas Appelt, Mabry Tyson, John Bear, and David Islael. SRI International: Description of the FASTUS system used for MUC-4. In *Proceedings fo the 4th Message Understanding Conference (MUC-4)*, pages 268–275, 1992.
28. Damaris Ayuso, Sean Boisen, Heidi Fox, Herb Gish, Robert Ingria, and Ralph Weischedel. BBN: Description of the PLUM system as used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 169–176, 1992.
29. Roman Yangarber and Ralph Grishman. NYU: Description of the Proteus/PET system as used for MUC-7 ST. In *Proceedings of the 7th Message Understanding Conference: MUC-7*, Washington, DC, 1998.
30. Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 811–816, 1993.
31. Stephen Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, 1999.
32. Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, July 2002.
33. Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 474–479. AAAI Press/MIT Press, 1999.
34. Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.
35. Ellen Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, 85(1-2):101–134, 1996.

36. Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany, August 2000.
37. Ion Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the International Conference on Autonomous Agents*, pages 190–197, Seattle, WA, USA, 1999. ACM Press.
38. Nick Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.
39. Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual Web Information Extraction with Lixto. In *Proceedings of the Conference on Very Large Databases (VLDB)*, 2001.
40. Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the International Conference on Autonomous Agents*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.
41. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the Conference on Very Large Databases (VLDB)*, pages 109–118, 2001.
42. Jan H. Van Bemmel and Mark A. Musen, editors. *Handbook of Medical Informatics*. Springer Verlag, Heidelberg, 1997.
43. Cincinnati Children's Hospital Medical Center. Evidence based clinical practice guideline for children with acute bacterial sinusitis in children 1 to 18 years of age. Cincinnati Children's Hospital Medical Center, Cincinnati, OH, April 27 2001.
44. Katharina Kaiser. *LASSIE – Modeling Treatment Processes Using Information Extraction*. PhD thesis, Institute of Software Technology & Interactive Systems, Vienna University of Technology, 2005.
45. Andreas Seyfang, Robert Kosara, and Silvia Miksch. Asbru 7.3 Reference Manual. Technical Report Asgaard-TR-2002-1, Institute of Software Technology & Interactive Systems, Vienna University of Technology, Vienna, Austria, Europe, 2002.
46. Wendy Lehnert, Claire Cardie, David Fisher, Joseph McCarthy, Ellen Riloff, and Steven Soderland. Evaluating an Information Extraction system. *Journal of Integrated Computer-Aided Engineering*, 1(6), 1994.