A Graphical Framework for Specification of Clinical Guidelines at Multiple Representation Levels

Erez Shalom B.S.c and Yuval Shahar M.D., Ph.D
Medical Informatics Research Center
Department of Information Systems Engineering
Ben Gurion University, Beer Sheva 84105, Israel
http://medinfo.ise.bgu.ac.il/medlab/
{erezsh, yshahar}@bgu.ac.il

Abstract *Formalization of a clinical guideline for purposes of automated application and quality assessment mainly involves conversion of its free-text representation into a machine comprehensible representation, i.e., a formal language, thus enabling automated support. The main issues involved in this process are related to the collaboration between the expert physician and the knowledge engineer. We introduce GESHER - a graphical framework for specification of clinical guidelines at multiple representation levels. The GESHER architecture facilitates incremental specification through a set of views adapted to each representation level, enabling this process to proceed smoothly and in a transparent fashion, fostering extensive collaboration among the various types of users. The GESHER framework supports specification of guidelines at multiple representation levels, in more than one specification language, and uses the DeGeL digital guideline library architecture as its knowledge base. The GESHER architecture also uses a temporal abstraction knowledge base to store its declarative knowledge, and a standard medical-vocabularies server for generic specification of key terms, thus enabling reuse of the specification at multiple sites.*

Introduction: Automated Tools for Specification of Clinical Guidelines

Clinical guidelines (GLs) have been shown to improve the quality of medical care, and are expected to assist in containment of its costs as well.

During the past 20 years, there have been several efforts to support complex GL-based care over time in automated fashion. This kind of automated support requires formal GL-modeling methods. Most of the methods [1-11] use knowledge acquisition (KA) tools for eliciting the medical knowledge needed for the knowledge roles (KRs) of the GL specification ontology (key concepts, properties and relations) that each method assumes, in order to specify it in a formal format. Using a recent definition [12], there are 2 main approaches to GL specification: document-centric, i.e., start from a free-text document and map it to a given GL ontology, and model-centric, i.e., model the GL de-novo using a predefined ontology and computational model, and refer to the source text only for documentation. We shall see that our architecture provide support for both approaches.

In most of those tools, however, the process of specification the GL into a formal language is not sufficiently smooth and transparent. The core of the problem involved in specification a large mass of free-text GLs into a formal machine readable format, is that expert physicians (EPs) cannot (and need not) program in GL specification languages, while programmers and knowledge engineers (KEs) do not understand the clinical semantics of the GL. In addition, some of the GL's knowledge is implicit and must become explicit during the specification process. The process should also support and facilitate iterative collaboration between these two different types of users - EPs and KRs. Finally, there is a need for a framework that deals with GLs represented in multiple ontologies.

The DeGeL Framework

The Digital electronic Guideline Library (DeGeL)[13], is a web-based, distributed architecture that embeds several web-based software tools. DeGeL supports GL classification, semantic markup, context-sensitive search, browsing, run-time application, and retrospective quality assessment. The DeGeL library supports multiple GL ontologies, in each of which, GLs represented in a hybrid format.

One of the functions the DeGeL architecture supports is the process of gravitating a set of GLs gracefully from text-based, to semi-structured text (labeled by the semantic KRs of selected target GL ontology), through semi-formal, down to formal,

machine-readable, executable representations. This incremental process is accomplished as follow: free-text GLs are loaded into DeGeL. EPs index and mark up the GL, using semantic labels from chosen target GL ontology (GL specification language), creating semi-structured representation, and in collaboration with a KR creating semi-formal GL representation. Then, KRs use an ontology-specific tool to add executable expressions in the formal syntax of the target ontology. Thus, each GL is represented in one or more representations levels: free-text, semi-structured text, semi-formal, and a fully structured representation. It is even possible to specify different KRs in the same GL using different representation levels (e.g., having fully structured eligibility conditions supports automated eligibility determination). All of those hybrid GL's representation levels co-exist and are organized in the DeGeL library in a unified structure - the hybrid GL representation.

The GESHER System

We have designed and implemented a new framework for GL specification in multiple representation Levels called ***GESHER***[1]. GESHER is a graphical client application, developed with Microsoft Dot.Net WinForm technology (see figure 1). GESHER supports the gradual specification process of the GL according to DeGeL's hybrid GL representation model which is not dependent on any particular medical domain. The GESHER system manages the specification process which incrementally structures a GL at multiple representation levels, according to the chosen target ontology. Each target ontology composed from KRs whose semantics are determined by the specification ontology (e.g., filter condition KR in the case of Asbru ontology)

During this incremental process, several of the GL's KRs might exist at different levels of specification. In addition, the GL itself might be decomposed into sub-guidelines that are candidate for further specification. The specification task is best handled within one integrated framework. Thus, GESHER uses DeGeL server (see figure 1) for managing, storing and accessing the relevant procedural knowledge at all levels during the specification process. GESHER uses graphical interface, enabling the gradual specification process be accomplished by an EP with relatively little training, by KE, or by both collaborating together. EPs are usually involved in the semi-structuring phase as they are familiar with the clinical knowledge inhered to the GL. KEs are usually involved with the semi-formal and formal phases as they are familiar with the GL specification language (the underlying target GL ontology). Sometimes, both EP and KE may work together (see figure 1), especially on the semi-formal level. GESHER provides specification tools for the different types of users in all stages of the specification process, allowing capturing the GL's semantics and enabling EPs and KEs to define its procedural and declarative knowledge. The task of using multiple ontologies in GESHER is supported by DeGeL's hybrid meta ontology[13], which supports specification of multi-ontologies of GLs, at least at the semi-structures level, using multiple target ontologies such as GEM or DeGeL's default ontology, Asbru. The meta ontology specifies the KRs common to all GL ontologies (e.g. documentation, classification indices) and the format in which any ontology is described. The knowledge editing authentication model is taken very seriously and is handled by DeGeL's authorization and authentication model- DeGeLock[13]. After a user is authenticated, her profile is retrieved. The user's profile contains only the tasks she is permitted to perform in GESHER (and implicitly on DeGeL's KB).

Some of the of the GL's specification is of a declarative type. For example, in the semi-structured level, an EP can find a description of what is "high HGB state" in a particular context. The concept "HGB State" is an example of declarative knowledge, which should be defined in a formal format according to its various allowed states: High, Normal or Low. For this task, or when needed to define complex expressions, GESHER is linked the temporal abstracted knowledge acquisition tool (TAKAT) (see figure 1), which is one of the tools used within the IDAN architecture [14]. For simple expressions (e.g. "HGB > 7.8 gr/dl"), the GESHER system uses the Expression Builder module. When EPs or KEs need to define a clinical term (e.g., "serum hemoglobin"), they can select standard term from a medical vocabulary. Standard terms are defined using several controlled medical vocabularies and can be searched and retrieved using the MEIDA[13] system (see figure 1), which includes a vocabulary server and a search engine.

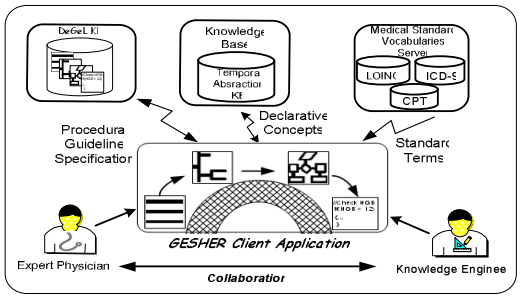

Figure 1. The GESHER Architecture

1 GESHER means " bridge" in Hebrew and stands for the bridge between the expert physician and the knowledge engineer in the process of guideline specification .

The use of standardized vocabularies and terms enables execution of queries in the IDAN framework by the Spock GL runtime application system [13], regardless of the terminology used in each local clinical DB. Using the highly intuitive, user friendly, graphically oriented interface of the GESHER system, the specification process becomes smooth for both the EP and the KE.

By developing specific graphical widgets for acquisition of each KR at each representation level, GESHER answers the needs of each one of the user types. For example, the methods needed by the EP specifying semi-structured knowledge (e.g., widget for markup of free text) are significantly different from the methods needed by the KE (e.g., widget for visual programming).

The GESHER Interface

The GESHER architecture integrates several tools. The main one is a graphical tool for guideline mark-up, editing and specification in multiple representation levels, which facilitates the collaboration between the EP and the KE. The main interface of GESHER system is presented in figure 2: with GESHER an EP can a new GL document (GLDoc) according to one of the target ontologies available in DeGeL (e.g., GEM, Asbru). Then, one or more guideline sources (GLSrc), are selected from the DeGeL GL library using the Vaidurya GL search and retrieve engine [13], and portions of it's free text are labeled as a starting point (see figure 2). The smooth process of specification is enabled by the *Hybrid Ontology Tree* (HOT).

HOT is composed from the KRs of the selected target ontology and has three different displayed views according to the three representation levels: semi-structured, semi-formal and formal view. The KRs which composing the HOT might be in different specification levels and are depending on the displayed view. For example, the KR *obtain values* is relevant to the semi-structured and semi-formal levels and therefore will be display when viewing those two views, while formal KRs such as Asbru's *returns* or *arguments*, are relevant to the formal level and therefore will be display in the formal view only. The different representation levels for every KR are implemented in the HOT as nodes and sub nodes. Each level is a sub-node of the parent KR node.

Thus, for the same KR, there are three different sub nodes, each node for each representation level. For each KR at a representation level, a graphical widget is generated on the fly (see figure 2). This widget is added to the upper *view* area or to the lower *design* area, enabling specification of the selected KR according to the selected representation level. Currently, we support the semi-structured level for multiple GL ontologies (e.g GEM, Asbru), and the semi-formal level for the default ontology used in DeGeL, Asbru. Thus, we have created the *Hybrid Asbru* ontology which embeds the semi-structured, semi-formal and formal Asbru semantics in one hybrid ontology.

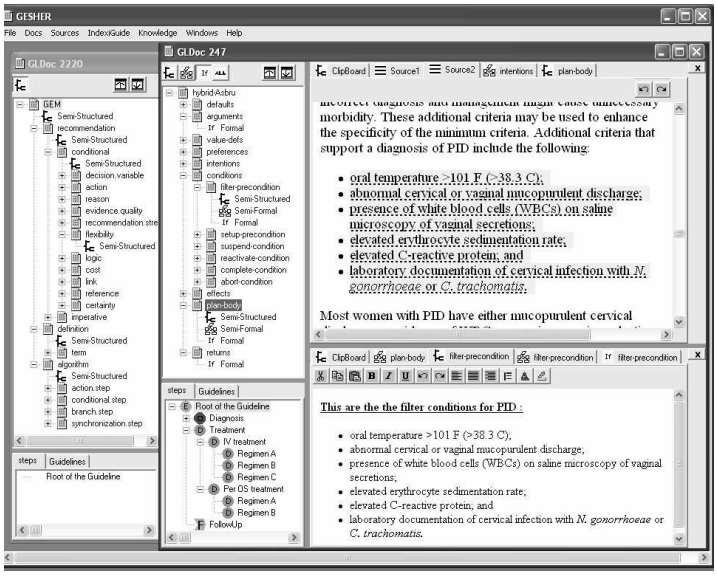

Figure 2. The GESHER system's main interface. Two guidelines documents are shown, each with a different target ontology: GEM (background window) and hybrid Asbru (front window). In the hybrid Asbru front document, the hybrid ontology tree appears in the upper left, showing the semi-structured, semi-formal and formal views. The guideline source is open in the upper view area and the semi structured filter-condition knowledge role (KR) (along with other KRs) is open in the lower design area, containing portion of the labeled text

The Semi-Structured Representation View

The semi-structured view is implemented as a graphical widget generated for any KR in the semi-structured representation level (in any target ontology). This widget contains HTMLEditor frame which enables the user to perform semi-structured mark-up by dragging a portion of labeled content from one or more GLSrcs (this content may be text, tables of figures) into a selected KR's frame, and perform manipulation on this text with reach designtoolbar. The positions of this markup at each GLSrc are saved and the labeled text is highlighted (see figure 2). This process enables turning implicit knowledge into more explicit fashion by the EP, facilitating the task of the KE towards formal level.

The Semi-Formal Representation View

Structuring GL at the semi-formal level requires developing designated widgets with intuitive interfaces to handle the acquisition accordance KR semantics. For example, we implemented a semi-formal view for semi-formal Asbru. Semi-formal Asbru is a simplified version of Asbru, with similar semantics to the full version, but with somewhat less complex syntax. The main reason we use semi-formal Asbru is to improve the collaboration between the EP and the KE during the GL specification process, especially after an EP semi-structured the GL and before a KE semi-formalized it. Semi-formal Asbru has most of Asbru's KRs such as plan-body which embedded the procedural knowledge of the GL and acquired with the Hierarchical Plan Builder Tool and conditions (e.g. eligibility completion, and filter condition) for describing time-annotations and simple temporal constraint which are acquired with the Expression Builder.

The Hierarchical Plan Builder

The semi-formal representation level is usually specific to each selected GL ontology, therefore, we have implemented the Hierarchical Plan Builder (HPB) (see figure 3), which is customized for the procedural aspects of the hybrid Asbru ontology (e.g., the plan-body KR in the case of the hybrid Asbru ontology). However, most of GL ontologies describe the procedural knowledge as a hierarchy of plans and sub plans, thus, this tool is actually quite generic. The HPB facilitates the task of decomposing the GL into sub-guidelines in a transparent process: The first step towards decomposing the GL into sub-plans is very straightforward and usually performed by the EP by specifying in general fashion the GL structure using an initial plans with basic semantic type (see figure 3). This type can be Intervention (e.g., *Education*, *Procedure*, or *Drug* types) or more general as *Observation* or *Follow-Up* types. In addition, we allow *General* plan, which its type will define later. For referring a pre-defined GL in DeGeL, *Public-Ref* plan can be created. When needed control elements, plans with semantics of *periodic*, *condition*, or *Switch-case* can be defined. Each plan is candidate for decomposing into sub-plans. When it decomposed, a new sub-level is created with at least two plans which are might further decomposed as well. In addition, control structure of ordering (e.g. sequential, parallel) might be added for each group of plans in the same level (see figure 3). A labeled portion of text with the plan's description can be related to each plan from its parent plan textual content. In the next step, which usually performed by the KE, or together with the EP, the user further specifies the plans and its sub-plans into semi-formal format (semi-formal Asbru in this case). For example, each plan with initial semantic type (e.g., drug, education) is *"To-Be-Defined"* plan in semi-formal Asbru. A decomposed plan specified as "*Subplan*", periodic plan as "*Cyclical*", and conditional plan as "*if-then-else*" in the semi-formal Asbru. Condition and switch-case plans have an expression as part of their semi-formal Asbru semantics. This expression can be specified in a semi-formal format using the Expression Builder). Thus, the HPB is a bridge between the EP and the KE, enabling collaboration towards the semi-formal and formal representation levels.

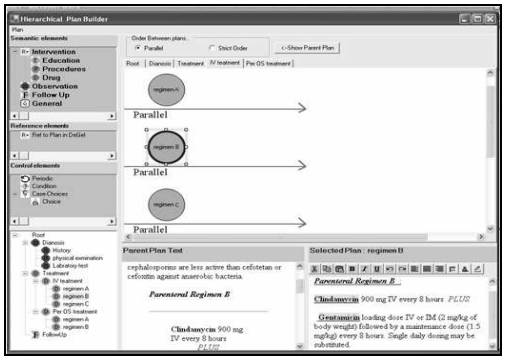

Figure 3. The Hierarchical Plan Builder in GESHER, showing how the procedural aspects for the guideline being specified. In this case, a plan for IV treatment of the Pelvic Inflammatory Disease, which specifies three different regimens that should be performed in parallel

The Expression Builder

Conditions (e.g., filter, complete) are one of the most important KRs in hybrid Asbru. A condition is a boolean expression which is evaluated to true or false. An example of using condition as filter condition is when trying to formalize the sentence "a pregnant woman whose hemoglobin level is greater then 20.gl/dl". The graphical widget used

within the GESHER system for defining conditions is called the "Expression Builder". IDAN language [14] used as the default formal query language.

The Formal Level Representation View

Our research focused mostly on the semi-formal level, which is the main phase in the GL specification process. After this phase most of the GL specification was defined and the KE can continue to the final step, the formal representation level. Nevertheless, work was done towards full-structured level: expressions, time annotations and simple temporal patterns which are conditions can be expressed with the Expression Builder and thus enabling execution in the IDAN by Spock system.

Discussion

We have presented GESHER, a client application graphical tool for specification of clinical GLs in multiple representations and in multiple GL ontologies. By using the GESHER tool, which is independent of any particular medical domain, the specification process of the GL becomes smooth and transparent, enabling collaboration between EP and KE. Using previously defined terms, GESHER can be characterized as both a *document-centric* and a *model-centric* tool, since it supports both a bottom-up ontology-driven semantic markup of the source document (mostly for declarative knowledge roles), as well as a top-down construction of the procedural body of the GL. GESHER uses DeGeL as the procedural knowledge server, allowing storage and re-use of knowledge that was acquired using graphical widgets which are generated for each knowledge role, for any representation level. Finally, temporal abstracted knowledge base enables declarative specification of the GL, and terms from standard medical vocabularies can be searched and use in the GL by using the MEIDA search engine.

An ongoing evaluation of the GESHER tool is currently being conducted as a joint study with collaborators in several clinical domains, with encouraging preliminary results. We have developed a methodology for the markup process and for evaluating the tool according to a detailed scaling and grading scheme. Future work on the GESHER framework will focus on the formal representation level. We aim to develop additional graphical widgets to support the specification of formal. In addition, we expect GL ontology designers to develop additional widgets to support the semi-formal, and formal views of other specification languages.

Acknowledgments

This research was supported in part by NIH award LM-06806. Drs. M. Goldstein, S. Martins, L. Basso, H. Kaizer, Prof. E. Lunenfeld, Drs. A. Yarkoni, and G. Bar were extremely helpful in the evaluation of all DeGeL tools.

References

[1] Ram P, Berg D, Tu SW et al. Executing Clinical Practice Guidelines using the SAGE Execution Engine. Medinfo. 2004;2004:251-5.

[2] Johnson PD, Tu S, Booth N, Sugden B, Purves IN. Using scenarios in chronic disease management guidelines for primary care. Proc AMIA Symp. 2000;:389-93.

[3] M. Peleg, A. A. Boxwala, O. Ogunyemi et al. GLIF3: The Evolution of a Guideline Representation Format. In: Proc. AMIA 2000.

[4] Shiffman RN, Karras BT, Agrawal A, Chen R, Marenco L, Nath S. GEM: a proposal for a more comprehensive guideline document model using XML. J Am Med Inform Assoc. 2000 Sep-Oct;7(5):488-98.

[5] Fox, J., Johns, N., and Rahmanzadeh, A., Disseminating medical Knowledge: the PROforma approach. Artificial Intelligence in Medicine, 1998. 14: p. 157-181

[6] Shiffman RN, Karras BT, Agrawal A, Chen R, Marenco L, Nath S. (2000). GEM: a proposal for a more comprehensive guideline document model using XML. Journal of the American Medical Informatics Association 7(5): 488-498

[7] Terenziani P., Montani S., Bottrighi A., Torchio M., Molino G., Correndo G. The GLARE Approach to Clinical Guidelines: Main Features. (CGP-04), Praha 2004

[8] Quaglini, S, Stefanelli, M, Cavallini, A, Micieli, G, Fassino, C, Mossa, C. Guideline-based careflow systems. Artificial Intelligence in Medicine 2000;5(22):5-22

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

[10]Kosara, R.; Miksch, S.: Metaphors of Movement: A Visualization and User Interface for Time-Oriented, Skeletal Plans, Artificial Intelligence in Medicine, pp. 111-131, 22(2), 2001.

[11] Votruba, P.: Structured Knowledge Acquisition for Asbru. Master's Thesis, Institute of Software Technology and Interactive Systems, Vienna University, Vienna, Austria, 2003

[12]Ružicka M., Svátek V. Mark–up based analysis of narrative guidelines with the Stepper tool. Proc. (CGP-04), Praha 2004

[13]Shahar Y, Young O., Shalom E, et al. A hybrid, multiple-ontology framework for specification and retrieval of clinical guidelines. Journal of Biomedical Informatics 2004, **37**(5), 325-344.

[14]Boaz, D., and Shahar, Y. (2005). A distributed temporal-abstraction mediation architecture for medical databases. *Artificial Intelligence in Medicine* **34** (1), 3-24.