Automated Generation and Deployment of Clinical Guidelines in the ICU

Femke De Backere, Hendrik Moens,
Kristof Steurbaut, Filip De Turck
Ghent University - IBBT,
Department of Information Technology (INTEC)
Gaston Crommenlaan 8, bus 201,
9050 Ghent, Belgium
Email: femke.debackere@intec.ugent.be

Kirsten Colpaert, Chris Danneels,
Johan Decruyenaere
Ghent University Hospital,
Department of Intensive Care
De Pintelaan 185,
9000 Ghent, Belgium
Email: kirsten.colpaert@ugent.be

Abstract

The complexity and amount of medical information and data keeps increasing, which makes it difficult to maintain the same quality of care in the Intensive Care Unit, without significant cost increases. In order to contain this complexity, clinical guidelines are used to structure best practices and patient care, but they also support physicians and nurses in the diagnostic and treatment process. Currently, no standardized format exists to represent these guidelines. Moreover, they are often handwritten. Translating guidelines into a computer interpretable format can overcome problems in their workflow and improve clinician's uptake. To this end, we developed an automated generation and execution engine. Based on the requirements, both functional and non-functional, an architecture using the microkernel pattern is presented. This allows us to easily add and modify functionality. This architecture was evaluated with the guideline for the calculation of calorie need for burn patients, used on a daily basis in the Intensive Care Unit of the University Hospital of Ghent.

1. Introduction

When taking medical decisions, physicians and nurses are often assisted by clinical practice guidelines (CPGs). These guidelines are descriptions of diagnostic processes and treatments. They can be handwritten or represented in an electronic format. CPGs can also be used when new processes and procedures are taught to the medical staff. The standardization of treatment, originated from the use of CPGs, ensures that patient care improves and minimizes healthcare expenditure. Therefore, the Institute of Medicine (IOM) has recommended the creation and use of CPGs since 1990. Despite this, no real standardization has occurred, leading to a multitude of different formats [21].

Moreover, the emphasis in medical institutions is more on the creation of guidelines, rather than on the use and practical implementation in a hospital setting [14]. Nowadays, clinical guidelines are often represented as plain text, which makes them difficult to handle. Additionally, it becomes more and more complex to teach such guidelines to physicians [18]. This also makes it inconvenient to search and execute the proper guideline. Guidelines increasingly are represented in a formal way, using digital guideline formats, also called computer-interpretable guidelines (CIGs). Often, these CIGs can be executed automatically.

A major issue is the number of available CIG formats that can be used to represent guidelines. Each format has its strengths and weaknesses. Moreover, it is not easy to convert one format into another, because, although these formats are very similar, they use other concepts and structures [16]. Another problem is that only little attention is paid to the execution speed of those guideline formats. Additionally, problems occur when clinical guidelines are translated manually into working computer applications. A platform for the dynamic composition of medical services can be used [8] to address this problem, but the communication gap between domain experts and software developers and lack of domain-specific knowledge from both sides [6, 7] still poses a problem. Other approaches exist that try to provide a solution to the known difficulties with guidelines. Some examine how existing CIG formats can be extended with evidence information to enrich the guidelines [15], others develop systems to transform text-based guidelines using a semi-automatic way based on natural language processing (NLP) [12].

Currently, medical decision support services are not distributed efficiently throughout the network. Some services need to be executed simultaneously and others use large amounts of data. Prior work on optimizations, such as distribution of services, is reported upon in [4], where the focus

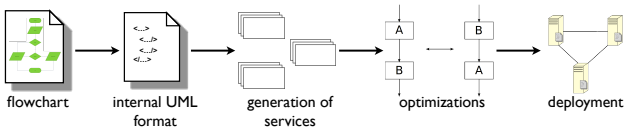

Figure 1. Overview of the functional requirements of the CPG execution engine

is on service execution and not on guidelines execution.

The Unified Modeling Language (UML) [19] is a standardized language, used to model different types of information using different diagram types. It should be familiar to most computer scientists. Although UML is a complex language, straight-forward diagrams can be interpreted by non-specialists. The UML activity diagram, which is used to model flowcharts, is of particular interest, as most CPG formats are flowchart-based. UML and more particularly state diagrams are already being used to verify the guidelines, by means of model checking [17].

A guideline execution engine can overcome the problems described in the previous paragraphs. This engine is able to translate a guideline, represented as a CIG format, into an internal UML representation. After this translation, the guideline can be optimized, followed by the automatic execution. The proposed architecture has to cope with a broad spectrum of similar standards and technologies. Moreover, it should be extensible and adaptable to easily integrate new optimization algorithms or new guideline formats. Therefore, a strict separation has to be made between fundamental, essential and additional functionality. Providing clinical decision support will benefit both patients and medical staff. It improves practitioner's performance [6], reduces medical errors [10] and minimizes costs and the length of patient's stay [1].

The remainder of the article is structured as follows. In Section 2 the functional overview of the engine is presented. Section 3 is devoted to the architectural details and implementation. A practical use case, the calculation of calorie needs for burn patients is demonstrated in Section 4. The evaluation of this use case will be discussed in Section 5. Finally, in Section 6 the main conclusions of this research are highlighted and possible directions for future research are described.

2. Functional Overview

The system should allow the user to enter a guideline into the system through a web interface or client application. This guideline can be represented in a format, specifically developed for this application, or as a CIG. In the latter case, the guideline is translated into an internal representation format used by the application. This format is based on

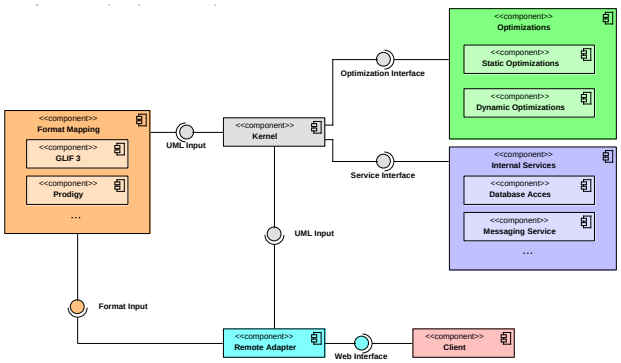

Figure 2. Overview of the CPG execution engine architecture

UML. When the translation is complete, the internal UML guideline components are converted into several services. This conversion translates decision steps and calculations automatically. Other services, such as messaging and data look up have to be predefined. In a next step, optimizations can be performed and finally the guideline is executed. An overview of these demands is illustrated in Figure 1.

In order to satisfy these functional requirements, the architecture has to be modifiable. This enables the translation of multiple CPGs formats (e.g. GLIF 3 [2] and Arden Syntax [11]). Such an approach also makes it possible to easily add new services or optimization techniques afterwards, without the need to stop the engine.

3. Architecture Details

In this section, the components of the architecture are described together with their interactions and implementation details.

3.1 Component Description

The high level architecture, as shown in Figure 2, is based on the microkernel pattern [3]. This pattern is mainly designed for systems in changing environments, which have to be extremely adaptable and extensible. To achieve this, a minimal, functional core is used, the kernel, which is split off from the rest of the system. This kernel works together with other components, which can easily be replaced or modified. The major advantage of using a microkernel is the increased reliability of the engine and its extensibility. The architecture consists of 6 packages.

The *Client* package is responsible for offering a simple user interface (UI). This can either be a standalone application or a web client connecting to a web application, offered by the Remote Adapter component. Diagrams can be fed to

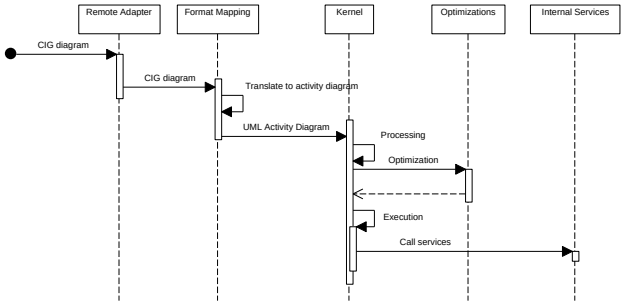

Figure 3. Overview of the interactions between the components of the CPG execution engine for the guideline generation and deployment

the kernel using the *Remote Adapter*. This can be done directly when diagrams are supplied in the internal UML format, or indirectly by offering them to the format mapping component, which converts the offered format to the internal UML representation. The Remote Adapter emulates the interface of existing CPG execution engines. This makes it possible for other applications to interact with the system.

As previously mentioned, the existing system is also capable of translating CIG formats into the internal UML guideline format. These conversions are handled by the package *Format Mapping*. For every CIG format that can be converted, a specific component is present in this package. By translating the diagram to an internal format, it is possible to translate different CIG formats just by dynamically adding new components to this package.

The *Kernel* package is the central component of the architecture. This package implements central services, such as handling communication and resources and only deals with the interpretation of diagrams at the highest level. This is done by using interfaces to other packages. The kernel is responsible for opening, interpreting and dynamically executing UML diagrams. The functionalities of the kernel are kept as small as possible.

The *Optimizations* package receives the diagram, executes the necessary optimizations and returns the result to the Kernel package. An example is the optimized distribution of the guideline. The *Internal Services* package extends the functionality provided by the kernel. This package provides the services needed to execute certain components within the activity diagram. Some examples are data lookup services, services providing communication with users and interaction with other components. More specific examples are discussed in Section 3.3.

Table 1. Tagged values and stereotyping

Stereotype name	Tagged value	Meaning
ServiceCallAction	Name	The class name of the service
	Parameters	Parameters that are passed to the service
ServiceCallAction	Name	RangeCheckService
	Parameters	age | [18, 30[
ServiceCallAction	Name	JEvalService
	Parameters	#{male}

3.2 Component Interaction

The interactions between the various components and packages are shown in Figure 3. This sequence diagram illustrates what happens when a CIG diagram is entered into the system:

1. The CIG diagram is entered in the system through the remote adapter.
2. As the diagram has to be converted into the internal UML format, the remote adapter passes the diagram to the Format Mapping package.
3. The Format Mapping package translates and converts the diagram to an UML activity diagram.
4. When the translation is complete, the internal representation is passed on to the Kernel.
5. The Kernel is responsible for the processing of the diagram. When this phase is completed, the diagram is sent to the Optimization component.
6. After the optimization, the diagram can be executed.
7. In a final step, tasks such as performing calculations and passing messages is entrusted to the internal services package.

3.3 Implementation Details

Currently, some services are already present in the Internal Services package. As some guidelines consist of multiple flowcharts, these have to be called at the right time. This can be done by using the *CallSubDiagramService*. The *JEvalService* is used to translate decision nodes and calculation nodes automatically into a service. This is achieved by using the JEval library[1]. Some results are printed to the

[1] `http://jeval.sourceforge.net/`

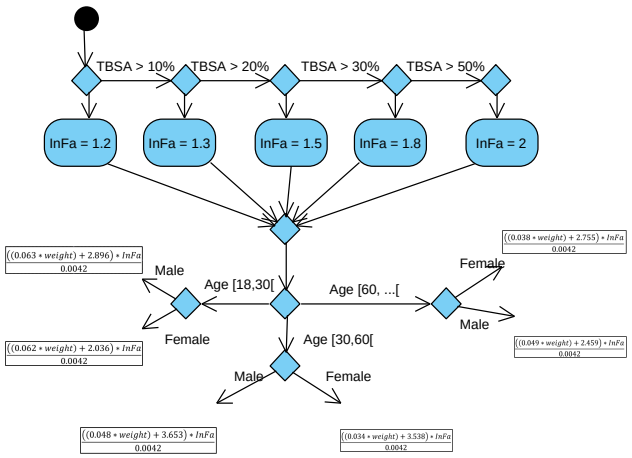

Figure 4. Guideline for the calculation of calorie need of burn patients

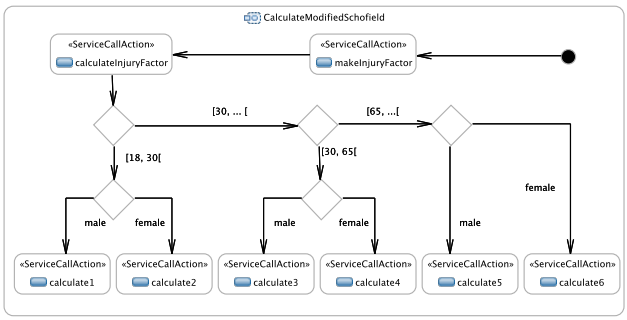

Figure 5. Internal UML representation

screen, using a simple *PrintService*. The *RangeCheckService* checks whether the value of the first parameter lays within the range of a second one. The *DataService* accesses the database to look up specific data.

The UML-based format was developed based on Eclipse MDT (Model Development Tools)[2], a free UML implementation. This format heavily uses profiles, tagged values and stereotypes. Tagged values supply additional information to model elements and add semantic information to the different modeling constructs. Stereotypes are the names that are added to the model element using «»-symbols. These stereotypes are defined within the profiles, while the stereotype themself contain the tagged values. By adding profiles to a certain diagram, the stereotypes can be accessed and used within the diagram. Adding a stereotype to a specific component enables the tagged values as additional attributes within the component.

Table 1 shows an example of both tagged values and stereotyping. The first entry introduces the general use of these concepts. The stereotype *ServiceCallAction* is used to call other services from the Internal Services package. It has 2 tagged values, *Name* and *Parameters*. The *Name* specifies the class name of the service that will be called, while the *Parameters* value determines the attributes that have to be provided to the service, seperated by "|". The second entry depicts how the RangeCheckService can be called, this is easily done by adding the name and necessary parameters to the ServiceCallAction stereotype. The name value will refer to the RangeCheckService residing in the Internal Services package. Two parameters are needed to activate this services, the *age* variable and the permitted range (e.g. [18,30[). The last entry shows the use of the JEvalService.

[2] http://www.eclipse.org/modeling/mdt/

Thus, the different UML nodes, through these stereotypes, contain information referring to specific services. These services are responsible for executing simple actions, such as the execution of calculations. As these services are referenced textually the existing system can easily be extended by adding new services.

4. Use Case: Calculation of Calorie Needs for Burn Patients

In this section, the need for and usage of the calculation of calorie needs for burn patients is discussed as well as the implementation by means of the above presented architecture.

4.1. Necessity

Ghent University Hospital is a tertiary care hospital in Belgium which has a 56-bed Intensive Care Unit (ICU). The 6-bed Burn Unit (BU) is a specialized part of the intensive care unit, where annually 130 patients are admitted. Patients who fulfill the specific requirements for admittance to the BU (i.e. severely burned patients, patients with inhalation injuries or patients requiring intensive wound treatment) are treated in this centre. One typical difference with other ICU patients is that burn patients have specific nutritional needs. As both underfeeding and overfeeding place an undesirable amount of stress on the body, it is important to correctly estimate their daily caloric needs [20].

Many research methodologies and results are vague about the optimal nutritional pattern for burn patients. However, there exists some estimations to calculate the daily caloric need [9]. Several parameters are taken into consideration: weight, length, sex and the total body surface area (TBSA). TBSA indicates what percentage of the body of the patient is burned.

Burn units in the United States and Australia experimented with formulas to calculate the nutritional need: the Harris Benedict equation [5], the Ireton-Jones formula [9]

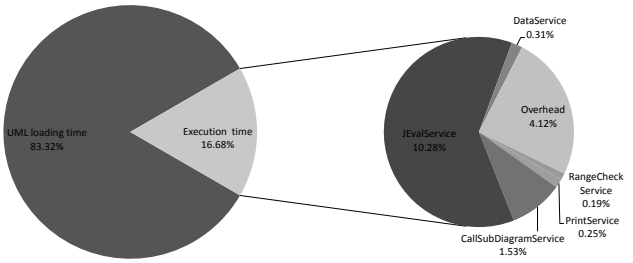

Figure 6. Breakdown of execution times

and the Modified Schofield formula [13]. These formulas are also used in the Burn Unit of the ICU in Ghent.

4.2. Modified Schofield and Implementation

Within the scope of this paper, the Modified Schofield calculation is used as an example (cfr. Figure 4). The Modified Schofield formula is divided into 2 parts. First, the injury factor is calculated. This value is based on the TBSA percentage. After the injury factor is determined, the calorie need is calculated, based on the sex, weight and age of the patient.

As a starting point, a flowchart of the Modified Schofield calculation was used. As a first step, this flowchart was turned into an UML activity diagram, as illustrated in Figure 4. This was a minor change and did not lead to any loss in legibility. Then, the various stereotypes and tagged values were added to the diagram.

Figure 5 illustrates the internal UML format for the Modified Schofield guideline and depicts the actions that are executed to determine the calorie need for a patient. In a first stage, the DataService looks up the necessary data. Next, the CallSubDiagramService calls the subdiagram to determine the injury factor of the patient. In this diagram, the if conditions are translated using the JEvalService. Subsequently, the actual calorie need of the patient is calculated. In a first step, the RangeCheckService is used to translate the range of the age. Next, the sex of the patient is determined, followed by the calculation of the calorie need, both these actions use the JEvalService. The results of the guideline are printed on the screen using the PrintService. All these services are called by using the ServiceCallAction stereotype.

5. Evaluation Results

The translation and execution of the Modified Schofield formula was evaluated to get more insights in the performance our application. Therefore, the average translation and execution time of the guideline to calculate the calorie need of 50 patients was tested. This test was repeated 20 times. This evaluation was executed on a desktop on an Intel Core 2 Duo T9400 2.53 GHz, OS Kubuntu 10.04. The Eclipse Modeling Edition was used to create the UML diagrams.

Table 2. Execution times for 50 patients

Part	Avg time (ms)	Std (ms)
UML loading time	2158.00	61.59
Execution time		
CallSubDiagramService	39.55	11.02
JEvalService	266.20	11.11
PrintService	6.55	2.93
RangeCheckService	4.85	1.87
DataService	7.90	2.85
Overhead	106.75	27.31
Total	2589.80	60.35

The results of analyzing the execution times are shown in Figure 6. This chart is divided into 2 parts. On the one hand there is the loading time of the UML library and the UML diagram, on the other hand, the execution time of the different services and the overhead caused by the microkernel pattern. An overview of the actual loading and execution times are depicted in Table 2. As shown in both Figure 6 and Table 2, the loading times for the UML library and diagram are most time-consuming. As this only has to be executed once, during the initialization of the application, this is not a major issue. The execution times of the services take very little time. Only the JEvalService consumes about 10% of the total execution time.

We can conclude that, provided that the UML library is initialized, execution of a guideline using our execution engine is very efficient. There still remains a relatively large overhead caused by the kernel. This is due to the dynamic loading of services.

6. Conclusion and Future Work

This paper describes how UML can be used in conjunction with clinical practice guidelines to resolve known problems with the translation of guideline formats into working computer applications. The Unified Modeling Language was chosen as an internal format because of its comprehensibility and interpretability. To this end, an execution engine was developed, which is capable of executing UML-based guidelines and mapping existing guideline formats on this new format. By using UML as an intermediate format on the one hand and by using the microkernel pattern on the other, various CPG formats can be executed on one engine. This reduces development and implementation cost as only

a new format mapping has to be added to the execution engine. The application of the microkernel pattern makes the engine easily extensible and adaptable. As a practical use case, the *Modified Schofield* flowchart to calculate the calorie need for burn patients was used. After an initial evaluation, we can conclude that the execution of the guideline is very efficient. The overhead is mainly caused by the dynamic loading of the services within the Internal Services package. The framework, presented in this paper, accelerates the translation and development of medical guidelines to model patient care and reduces the communication gap between domain experts and software developers.

Future research will focus on the further development of additional services, such as, implementing the Format Mapping package and additional services. Next, we will investigate specific optimizations for a selected set of ICU use cases, in order to improve the performance and reliability.

References

[1] J. Barenfanger, M. Short, and A. Groesch. Improved antimicrobial interventions have benefits. *Journal of Clinical Microbiology*, 39(8):2823–2828, August 2001.

[2] A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q. Zeng, D. Wang, V. Patel, R. Greenes, and E. Shortliffe. Glif3: A representation format for sharable computer-interpretable clinical practice guidelines. *Journal of Biomedical Informatics*, 37(3):147–161, June 2004.

[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-oriented Software Architecture - A System of Patterns*. John Wiley & Sons, Chichester, 1996.

[4] F. De Turck, J. Decruyenaere, P. Thysebaert, S. Van Hoecke, B. Volckaert, C. Danneels, K. Colpaert, and G. De Moor. Design of a flexible platform for execution of medical decision support agents in the intensive care unit. *Computers in Biology and Medicine*, 37(1):97–112, January 2007.

[5] R. Dickerson, J. Gervasio, M. Riley, J. Murrell, W. Hickerson, K. Kudsk, and R. Brown. Accuracy of predictive methods to estimate resting energy expenditure of thermally-injured patients. *Journal of Parenteral and Enteral Nutrition*, 26(1):17–29, January 2002.

[6] A. Garg, N. Adhikari, H. McDonald, P. Rosas-Arellano, P. Devereaux, J. Beyene, J. Sam, and B. Haynes. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: A systematic review. *Journal Of the American Medical Association*, 293(10):1223–1238, March 2005.

[7] M. Goldstein, R. Coleman, S. Tu, R. D. Shankar, M. O'Connor, M. Musen, S. Martins, P. Lavori, M. Shlipak, E. Oddone, A. Advani, P. Gholami, and B. Hoffman. Translating research into practice: Organizational issues in implementing automated decision support for hypertension in three medical centers. *Journal of the American Medical Informatics Association*, 11(5):368–376, September 2004.

[8] A. Hristoskova, D. Moeyersoon, S. Van Hoecke, S. Verstichel, J. Decruyenaere, and F. De Turck. Dynamic composition of medical support services in the icu: Platform and algorithm design details. *Computer Methods and Programs in Biomedicine*, May 2010.

[9] C. Ireton-Jones and J. Jones. Improved equations for predicting energy expenditure in patients: The ireton-jones equations. *Nutrition in Clinical Practice*, 17(1):29–31, February 2002.

[10] R. Kaushal, K. Shojania, and D. Bates. Effects of computerized physician order entry and clinical decision support systems on medication safety: A systematic review. *Arch Intern Med*, 163(12):1409–1416, June 2003.

[11] S. Kim, P. Haug, R. Rocha, and I. Choi. Modeling the arden syntax for medical decisions in xml. *International Journal of Medical Informatics*, 77(10):650–656, October 2008.

[12] P. Martini, K. Kaiser, and S. Miksch. Easing the formalization of clinical guidelines with a user-tailored, extensible agile model driven development (amdd). In *2008 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 120–125. IEEE, June 2008.

[13] B. Masters and F. Wood. Nutrition support in burns – Is there consistency in practice? *Journal of Burn Care & Research*, 29(4):561–571, July 2008.

[14] S. Maviglia, R. Zielstorff, M. Paterno, J. Teich, D. Bates, and G. Kuperman. Automating complex guidelines for chronic disease: Lessons learned. *Journal of the American Medical Informatics Association*, 10(2):154–165, March 2003.

[15] A. Ozturk, K. Kaiser, P. Martini, and S. Miksch. Embedding the evidence information in guideline representation languages. In *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)*, pages 512–517. IEEE, June 2007.

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

[17] B. Pérez and I. Porres. Verification of clinical guidelines by model checking. In *CBMS '08: Proceedings of the 2008 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 114–119, Washington, DC, USA, 2008. IEEE Computer Society.

[18] S. Quaglini and P. Ciccarese. Models for guideline representation. *Neurological Sciences*, 27(0):s240–s244, June 2006.

[19] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004.

[20] M. Suri, V. Dhingra, S. Raibagkar, and D. Mehta. Nutrition in burns: Need for an aggressive dynamic approach. *Burns*, 32(7):880–884, November 2006.

[21] D. Wang, M. Peleg, S. Tu, A. Boxwala, R. Greenes, V. Patel, and E. Shortliffe. Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines: A literature review of guideline representation models. *International Journal of Medical Informatics*, 68(1-3):59–70, December 2002.