The Practice of Informatics

**Progress with Formalization in Medical Informatics?**

**Abstract** The prevailing view of medical informatics as a primarily subservient discipline in health care is challenged. Developments in both general informatics and medical informatics are described to identify desirable properties of modeling languages and tools needed to solve key problems in the application field. For progress in medical informatics, it is considered essential to develop far more formal modeling languages, modeling techniques, and tools. A major aim of this development should be to expel ambiguity from concepts essential to medicine, positioning medical informatics "at the heart of health care."

**J Am Med Inform Assoc.** 2001;8:126–130.

Medical informatics researchers are urged to develop safe, accurate, and effective computer applications. They are often confronted with questions about whether a particular implementation has been fully realized or evaluated for major effects, even on such remote criteria as quality of life of patients. These are valid questions as they refer to milestones of the software-engineering life cycle, and, ideally, researchers would like to reply that an application has gained broad acceptance and has resulted in significant improvement of both health care and health. This is the ultimate goal of research into health care, and the following quote from the end of a thorough discussion on the nature of medical informatics is of interest in this respect: “The goal of medical informatics is to help health care workers improve outcomes, not to further the discipline itself.”

We would like to challenge this too-restricted view of the goal of medical informatics, because far too much emphasis is placed on the subservient aspects of (medical) informatics.

On the one hand, the above postulate assumes the existence of adequate methods, techniques, and tools for solving information-related problems in contemporary health care. Solving such problems then simply boils down to selecting the right methods, techniques, and technology and applying them correctly. Examples include the use of object orientation, component-based development, workflow modeling, conceptual graphs, graph grammars, and decision
networks.\textsuperscript{7} These are all, in a sense, solutions to restricted and well-defined problems, whose complexity is still manageable.

On the other hand, in the development of complex dedicated applications (such as configurable patient record systems for both care delivery support and research purposes, patient-record-based reviewing systems, predictive data entry systems using controlled vocabulary, care process monitoring systems, patient case simulators, and case report retrieval), this fundamental assumption is incorrect, and existing technology is often insufficient or unsuitable. Notice that these include, more or less, the top four of Sittig’s “grand challenges for medical informatics,”\textsuperscript{8} as expressed in the early 1990s. It might be argued that developments in health care are rather slow, that health care does not yet seem to require sophisticated applications that try to solve these kinds of complex problems, and even that existing solutions to simple problems are not yet widely used in clinical practice. In our opinion, this situation is, however, radically and rapidly changing.

First, a number of “simple” applications, like the systems in laboratories, signal processing departments, pharmacy departments, and general practice, are in fact widely used. In the Netherlands, for example, more than 90 percent of general practitioners use electronic treatment records and electronic mail, which are used to record current medical notes and to maintain problem lists.

Second, the growing pains experienced with these general-practitioner systems—i.e., their lack of functionality for the support of a lot of domain-specific work—were largely due to the fact that they originated in systems designed primarily to support clerical work. Recently, the clerical support functionality has been removed from the “classical” general-practitioner system and replaced by generic and integratable software packages from outside health care. This has left the harder-to-solve domain-specific support problems, like the definition and efficient use of a medical record, to the developers and vendors of the general-practitioner systems.

These vendors for some years now have failed to come up with a good next generation of medical record systems that meet the needs of users, and they have spent millions of dollars on projects that have been terminated prematurely because they have not had the proper domain-specific techniques and languages to build systems that really support the core of the general practitioner’s work! This tendency can also be observed in hospital information systems, in which open architectures enable the integration of “best of breed” solutions for such problems as appointment and resource scheduling, which have been developed outside health care (e.g., enterprise resource planning and human resource management solutions from industry), leaving the much harder-to-solve problems to developers and researchers in medical informatics.

In medical informatics, the improvement of existing methods and techniques and the invention of new ones has thus become a necessity. This is natural for any young discipline, and the acceptance of this fact should lead to the recognition of medical informatics as a field of study in its own right. Research in medical informatics will still, of course, have to lead to improved health care, but the focus is on the long rather than the short term.

To prevent what happened before in informatics, when solving a problem often meant inventing “yet another method or technique” (Avison and Fitzgerald even introduced the term “methodology jungle”\textsuperscript{9}), it is appropriate to look at some salient developments in informatics methods and techniques and to discuss requirements for methods and techniques to solve problems specific to medical informatics.

**Some Salient Developments in Informatics**

For real progress, the development of domain-specific modeling languages is imperative. In informatics, a general awareness has emerged that there is no universal method or “silver bullet”\textsuperscript{10} to solve all developmental problems—the development of specific modeling languages is common practice, for example, in the field of information systems.\textsuperscript{11,12} It is now obvious that developers of real-time systems, decision support systems, and knowledge-based systems have to use entirely different modeling techniques.

To illustrate the idea of domain-specific modeling languages, we first cite an example from the field of compiler construction. In the programming of a parser for compiling purposes, assembly code could be used to achieve all parser functionality. However, programming a parser in assembly language was, as an engineering process, very cumbersome and prone to error.

Clearly, what was needed was a language that directly supported concepts specific to parsers, and from this arose the fundamental notion of a grammar integral to such a language. This realization led to the development of such languages as CDL (Compiler Definition Language)\textsuperscript{13} and such tools as Lex\textsuperscript{14} and Yacc,\textsuperscript{15} which supported the new generation of parsers.
Compiler construction, as a field, has matured, and sophisticated compiler–compiler techniques have proved their value in practice. The development of complex compilers has also been greatly facilitated; for example, most parsers can be described in just a few pages and are also used now by domain experts in such fields as cognitive science and computational linguistics.

More examples of typical developments in informatics can be given. Concentration on the syntactic and semantic aspects of data modeling in early proposals of the entity-relationship model\textsuperscript{16,17} made it possible to set apart the problems of physical data storage and access. This gave domain experts the opportunity to do higher level modeling of the relevant aspects of their domain. Recently, in the field of workflow management systems, task coordination data have been separated from the actual applications, again giving domain experts specialized techniques for the real modeling of coordination.\textsuperscript{18}

It is important to understand that these developments were achieved only by means of a paradigm shift, which involved the development of domain-specific tools and techniques. It should also be understood that this paradigm shift reveals problems of ambiguity in general and semantic problems in particular. Finally, concepts in the successful languages in general match “naturally” with concepts in the application domains.

**A Parallel in Medical Informatics?**

A similar paradigm shift is needed in medical informatics, leading to the development of domain-specific languages that can be used by domain experts who are not necessarily computer experts. This need is more or less implicit in recent articles dealing with the development of reusable ontologies and clinical terminologies, for example.\textsuperscript{19–22} These articles also explore certain requirements, or desiderata, for methods and techniques for domain-specific modeling languages. Elsewhere in this journal,\textsuperscript{23} we detail a set of requirements for medical modeling languages.

What are domain-specific languages, and why is their development essential both in medicine and in medical informatics?

A domain-specific modeling language is a language specifically designed to solve specialized problems in a predetermined domain. The term “domain-specific,” as used here, refers to a level of specialization of the concepts used in the technique. Concepts in the problem domain match naturally with concepts in the language. Development of domain-specific modeling languages in the medical domain happens rarely. The ARDEN-syntax, for example, is domain specific with respect to modeling valid alert tasks. Alert tasks are triggered in special situations and produce logical messages to suggest actions. Hence, the medical logic modules (MLMs) of ARDEN\textsuperscript{24} use concepts like “evoke,” “logic,” “action,” “message,” and “literature citation.”

Developing this type of domain-specific languages provokes fundamental research questions, such as:

- What is the abstract structure of alert tasks and of medical tasks in general?
- What is the abstract structure of case reports?
- What formal semantics can be assigned to disease descriptions?
- What formal semantics can be assigned to a care coordination protocol?
- What formal relationship exists between disease course descriptions and individual patient histories?
- What formal relationship exists between review tasks and patient data?

Answering these questions involves “high-level modeling,” which has as a logical consequence “high-level tooling.” Some aspects of this high-level development are already visible in health care.

Currently, we observe that hospitals try to use a combination of document workflow techniques, document management techniques, and electronic mail to communicate orders in a very pragmatic way. This integration is still focused mainly on the level of software packages. It would benefit greatly from the availability of dedicated modeling techniques that could be applied throughout the organization at the conceptual level of order communication.\textsuperscript{25}

An example of a modeling technique especially developed for medical purposes is GRAIL.\textsuperscript{26} GRAIL is a modeling technique dedicated to classifying large numbers of medical concepts. GRAIL is, however, still a generally applicable modeling technique. The application of GRAIL in the development, for example, of a classification of medical procedures requires a very efficient high-level tool that can deal with the problems of scalability.\textsuperscript{27}

We further mention Disease Course Graph Language (DCGL\textsuperscript{28}) as a good example of a dedicated medical modeling technique. This language, which assigns formal semantics to concepts like “prodrome” and “course remission,” is specifically designed to model disease courses.
A good example of a required dedicated medical modeling technique is a technique to model electronic medical records, a.o., to deal with the differences across medical specialties. First steps in a fundamental treatment of the connected problems have been taken.\(^{29}\) Since there is no silver bullet medical record structure serving all requirements from various specialties and care settings (e.g., inpatient, outpatient, and shared care), a medical record modeling technique could be useful to communicate all kinds of medical record instances, which facilitates discussion and implementation of medical record solutions.

Other aspects of the use of medical records still require analysis. For example, medical records have to play an essential role in the exchange of results of clinical reasoning in settings in which clinicians share responsibilities for a patient. So the modeling technique should facilitate the modeling of the results of clinical reasoning at various intermediate stages of this clinical problem-solving process. For that, we have to know what can sensibly be said about these intermediate results of clinical reasoning themselves. Only initial informal descriptions seem to be available yet, and these will have to be thoroughly analyzed before they can be approached in a more formal way.\(^{30}\)

**Conclusions**

All the issues raised in the preceding paragraphs touch on questions that lie at the heart of health care and medicine,\(^{31}\) since they are basically concerned with expelling ambiguity from concepts essential to medicine. In fact, these questions can be completely uncoupled from the subject of computer implementations. Thus, we have to ask ourselves whether answering these questions can still be considered medical informatics? If so, medical informatics is positioned at the heart of health care and can be addressed any time medical information is ill understood or ambiguous.

In its early years, informatics as a discipline was defined both narrowly, like computer science, and broadly, like information science.\(^{32,33}\) If medical informatics is considered an information science, it is clear that questions that concern removing ambiguity of medical information are, indeed, one of its central subjects. But a more pragmatic argument can also be found: If information analysts are confronted with questions to which domain experts cannot yet give clear answers, medical informatics as a science will be forced to seek unambiguous answers itself, or else the specialized computer programs being sought will lack an adequate theoretic foundation and hence remain ad hoc.

Disambiguating medical information is, in fact, a form of conceptualization\(^{34}\) that benefits from the development of domain-specific languages, as language development concerns the definition of unambiguous syntax and semantics of the involved medical concepts to be represented. At this point, we are back where our discussion started, at the development of domain-specific languages. On the one hand, language development fosters the loss of ambiguity, while on the other hand, the aim to lose ambiguity promotes language development. If this dual process of conceptualization and proper definition of domain-specific languages can be referred to as “formalization,” then in our opinion the road to scientific maturity for medical informatics is through formalization. This means that the answer to the ambiguous question in the title of this viewpoint—Is there progress with formalization in medical informatics?—is “yes,” regardless of how the question is interpreted!

This view of formalization in medical informatics has major implications. It requires research into new topics, such as requirements of domain-specific languages (as we discuss elsewhere in this journal\(^{23}\)), education in applying formal techniques, and CASE-tool development. The scientific and educational institutions in medical informatics will play a key role in achieving progress through formalization. They will have to train the future generation not just in sound engineering practices but, equally important, in the formalization and application of proper tools, avoiding the pitfalls of *l’art pour l’art*, i.e., formalization without added value. They will have to discuss specifically the teachability (or comprehensibility) aspects of modeling languages.

**References**


