

Can We Formally Specify a Medical Decision Support System?

Paul Krause, John Fox, Mike O'Neil, and Andrzej Glowinski
Imperial Cancer Research Fund

COMPUTER-BASED DECISION support is finding increasing application in specialized medicine, but until recently there has been little progress in addressing primary care. General practitioners are facing more and more pressures that could severely limit their effectiveness. For example, as the breadth, depth, and complexity of medical knowledge increases, there is a corresponding increase in the gap between what a doctor should know and what can be humanly learned, retained, and applied. Motivated by this, the Imperial Cancer Research Fund, London, initiated the Oxford System of Medicine project.¹ The goal of the project is a decision support system (DSS) that general practitioners can use during routine clinical work to support such decision-making tasks as diagnosing illnesses, planning investigations and patient treatment schedules, prescribing drugs, screening for disease, assessing the risk of a particular disease, and determining whether to refer a patient to a specialist.

Work on the project to date has considered the design of a *general* DSS that can answer queries about any aspect of medicine.² However, we also must address the specification and verification of *medical* DSSs. What aspects of the system should we aim to address? We could apply the KADS methodology, which was developed

TO GAIN WIDESPREAD ACCEPTANCE, MEDICAL DECISION SUPPORT SYSTEMS MUST BE BUILT TO THE HIGHEST POSSIBLE STANDARDS. SYSTEM BUILDERS CAN EXPLOIT A VARIETY OF FORMAL TECHNIQUES TO ACHIEVE SUCH STANDARDS.

to bring a more structured approach to the development of knowledge-based systems. However, KADS is primarily targeted at developing models of expertise, which are but one component (although a central one) of a complete functional system. We believe that the development of a *functional* knowledge-based system requires a more holistic approach in which the system's intended environment plays an important role. That is, we need to draw formal specification techniques from software engineering into the life cycle of knowledge-based systems.

The need for a formal specification

Let's use an informal legal liability study to demonstrate why we should formally

specify knowledge-based systems. A system is in a potentially safety-critical situation if its failure or incorrect behavior might lead to loss or injury to a third party. By this token, many medical DSSs will be safety critical in their application. Consequently, the builders and suppliers of such systems are under a moral, and possibly a legal, responsibility to ensure that their systems present the best possible advice, reliably and in an unambiguous format. Medical DSSs will not gain widespread acceptance unless they can be demonstrated to "know" as much and perform at least as reliably as a skilled practitioner working in the same domain. That is, they must be able to demonstrate their professional integrity. If a third party suffers loss or injury as a result of following the advice of a DSS, the party may try to seek compensation from the builder or vendor. What steps can DSS

builders take to limit their liability in such cases?

In the sound engineering of conventional procedural software, we can legitimately strive for the ideal: that deploying the software will involve negligible risk. However, in the domain of medicine, where the current state of science precludes the development of a fully deterministic model, there will almost always be an uncertainty attached to a diagnosis, prescription, or referral. Consequently, a degree of risk will be associated with acting on the advice of a DSS no matter how carefully the system has been constructed. The best we can hope for is to minimize the risk when the system is used in a well-defined and explicitly specified context. The law recognizes this in imposing a "duty of care" on suppliers and users, not guarantees of safety in all circumstances. The DSS builder must use "appropriate standards of care" in designing and constructing the system. Unfortunately, it is not yet clear what constitutes an acceptable standard of care, or whether the imposition of such standards might circumscribe unacceptably the creativity of the system builder.

We can gain some insight into the requirements of such standards of care by taking a closer look at the circumstances in which potential liabilities might arise;⁴ that is, if

- (1) an expert system furnishes incorrect information;
- (2) the user unjustifiably relied on the information supplied by the DSS; or
- (3) an expert system was not consulted in a situation where it was standard practice to do so.

Circumstance 3 lies outside the scope of this article, being the subject of emerging codes of practice in those fields where expert systems are likely to have a significant impact. Circumstance 2 reminds us that a DSS cannot be designed in isolation from knowledge of its intended external environment. The context in which the system may be used should be clearly specified. Circumstance 1 can arise if the knowledge base contains erroneous information, or if erroneous inferences are drawn from the information contained in the knowledge base. Delving deeper, erroneous inferences might be drawn if

- (1) the inference engine is incorrectly

implemented or behaves in a poorly understood way;

- (2) correct data has been presented to the user, but in a misleading format;
- (3) a misleading user interface results in the user inputting incorrect information; or
- (4) the instruction manual or on-line help facility is unclear or misleading, resulting in incorrect use of the system.

A party who suffers harm from the use of a DSS has three theories on which to base a legal application for remedy:⁴

***BY STRUCTURING THE
SYSTEM INTO FUNCTIONALLY
DISCRETE SEGMENTS, WE CAN
DEVELOP DISTINCT MODELS
FOR DIFFERENT TASKS (USING
DIFFERENT TECHNIQUES
IF NECESSARY).***

- (1) negligence;
- (2) strict liability in tort; and
- (3) breach of contract/warranty.

Even though strict codes of practice might have been followed in developing a product, a defect that makes the product unreasonably dangerous can be grounds for liability in tort (that is, a wrongful injury to a person, a person's reputation, or a person's property). However, if a DSS is regarded as a service, there is as yet no precedent for applying liability in tort: Unless a system controls instrumentation, it is unlikely that physical injury will arise directly from any action of that system. This leaves negligence and breach of contract/warranty as the relevant theories on which to base a legal case. Defense against these theories lies, respectively, in the supplier using appropriate standards of care, and being able to demonstrate the fitness of purpose of the DSS.

Thus, we have two top-level requirements that we could help fulfill by extending formal specification techniques to all aspects of knowledge-based systems. First, specification of the full system

should include a clear and unambiguous statement of the intended purpose of the final system. Second, the specification technique should enable the developer to state clearly the behavior of the system in its intended environment, and should have associated with it a method for generating software that demonstrably conforms to the specification.

The role of formal specification

Exactly what constitutes a specification of a medical DSS? To meet the requirements of our earlier informal study of legal liability, a specification should cover at least the following:

- (1) top-level requirements, including the extent of the medical domain covered, the depth of understanding of the domain, and the level of professional competence of the intended user;
- (2) the decision-making tasks that the system is intended to support;
- (3) the semantics of the inference engine and any constraints this may impose on the rules of inference and the information content of the knowledge base;
- (4) the user interface; and
- (5) the information that should be present in the knowledge base.

Existing formal approaches to software development may help us here in specifying and developing the inference engine and the user interface. Techniques are also well developed (for rule-based systems) to enable the static analysis of the knowledge base.⁵ However, while important, such static analyses are limited in what they can achieve. They may guarantee that the knowledge base satisfies certain conditions to ensure the reasonable and efficient behavior of the inference engine. But, in order for the system to give the best possible advice, we must also ensure that the system has available to it and uses all and only the information that is relevant in a given decision-making context. How can we enrich our understanding of the intended and expected behavior of a DSS as a complete entity?

In many engineering disciplines, it is common for a model to be constructed as part of the process of specifying and

designing a system. This lets developers make predictions about the properties of the end product and analyze its behavior under extreme conditions before beginning construction. The model might be a physical scale model or a mathematical model. Both permit the design to be evaluated in terms of some known underlying theory. In the development of conventional procedural software, formal specification languages are intended to enable software engineers to produce a mathematical model of the software they wish to build before committing to coding it. As well as providing an unambiguous statement of the program's intended behavior, the formal specification may be subject to mathematical proofs and formal analysis of its properties. In addition, once the formal specification has been finalized, the developer can use stepwise refinement to generate software that correctly implements the specification.

Our approach

Although model-based specification languages such as Z⁶ or VDM⁷ (for Vienna Development Method) may have application to certain aspects of a knowledge-based system, a number of other modeling techniques can be applied; one paradigm is not necessarily applicable to the specification of the system as a whole. For a complex system, it might be more appropriate to consider the DSS a set of models.⁸ This is the approach we have taken in developing the Oxford System of Medicine. By structuring the system into functionally discrete segments, we can develop distinct models for different tasks (using different techniques if necessary). We have decomposed the OSM into the following modules:

- a user model;
- models of the decision-making tasks;
- an inference layer; and
- top-down development of the medical knowledge base using domain models.

To satisfy the requirements of a DSS destined for the medical generalist, the body of specific medical facts required for the OSM must have a very large medical content, representing the work of many

authors.¹ We must ensure that this knowledge base is as correct and complete as possible, and consistent across the contributions of many individuals. We have addressed these difficulties by developing an abstract medical knowledge model that deals primarily with the static part of the knowledge base, defining the types of objects this contains and their interrelations, along with specialist domain theories that help to maintain the integrity of the knowledge.⁹ More subtle are requirements for the knowledge to be represented at the expected level of detail, and for the

THE MODEL SPECIFIES WHAT IS REQUIRED IN THE KNOWLEDGE BASE, BUT IT DOES NOT COMMIT TO ANY IMPLEMENTATION, NOR DICTATE HOW COMPLIANCE SHOULD BE ACHIEVED.

degree of completeness demanded by the task specifications.

The model provides a specification of what is required in the knowledge base, but it does not commit to any implementation, nor does it dictate how compliance should be achieved. We have done some preliminary work on placing a more formal object-oriented structure on the OSM knowledge model using an object-oriented variant¹⁰ of the formal specification language Z. This notation provides a framework for formally specifying and reasoning about the behavior of user-defined classes of abstract objects. As with Z, data types are specified using simple set-theoretic constructs, with constraints on their values specified using first-order predicate calculus. A simple syntactic structure using schemas provides a framework for the specification. To introduce a specific class of abstract objects, the notation uses a state schema, which contains a characterization of the internal state for each instance of the class (state components together with state invariants), together with the conditions that must hold in any initial state. A state schema has the following generic form:

$\Gamma(\pi_1, \dots, \pi_k)$
 X Component declarations
 Y Invariant predicates
 Z Initialization predicates

The header of a state schema provides an identifier for the class in question (Γ in this case) and names any formal parameters for that class (π_1, \dots, π_k).

In keeping with the object-oriented approach, we can specify a refinement of a class that will inherit all the attributes of the parent class, but also contain additional components and be subject to additional state invariants and initialization conditions. In the usual application of such state schemas, the abstract objects referred to are states of the software system being specified. Event schemas are then defined, which describe the events that the state may undergo. However, we can use the same notation to describe objects that are components of the knowledge base. For example, we might use the following (simplified) schema to represent the abstract class Pathology (for pathological processes) in a medical knowledge base:

Pathology
 Symptoms : set Entity
 Signs : set Entity
 Contra-signs : set Entity
 Causes : set Disease
 $(\text{Symptoms} \cup \text{Signs}) \cap \text{Contra-signs} = \emptyset$

That is, in this simplified example, a pathological process has associated symptoms, signs, contrasigns, and causes. In any instance of the class Pathology, no element of the set of Contra-signs may also be a Sign or a Symptom. In the schema, the first three components are sets of an as yet unspecified data type called Entity. Entity may be declared as a given set, with no particular limitation on its elements. For example, Entity might be a set of text strings, in which case Symptoms, Signs, and Contra-signs are merely sets of labels with no additional information about the concepts they represent. Alternatively, we might use a more complex construction of this data type, constraining the properties of its elements in a clearly specified way. In this example, the causes of a pathological process are those diseases that have a known causal relationship with the pathology. To allow a more complex structure of the model, we could define the elements of

Causes to be instances of a class Disease, where Disease is itself specified as an object.

Consider, now, the more specific case of Arthritis as a refinement of the class Pathology:

Arthritis
Pathology
{Joint Pain, Joint Swelling, Joint
Tenderness} \subseteq Symptoms
{Joint Effusion, Joint Stiffness}
 \subseteq Signs

In this case, Arthritis inherits all the attributes of the class Pathology, but it has specific symptoms and signs associated with it.

A specific pathological process may also be an instance of a subclass of Pathology, with additional attributes from a class of physiological disorders. This example of multiple inheritance is easily accommodated in this notation:

Myocardial Ischemia
Ischemia
Cardiological Disorder
{Kawasaki Disease} \subseteq Causes

That is, Myocardial Ischemia inherits all the attributes of a localized deficiency of blood (Ischemia) in the specific context of a cardiological disorder.

This approach to modeling the knowledge base should also have benefits for knowledge elicitation. For example, in defining Acute Myocardial Ischemia as a refinement of Myocardial Ischemia, the expert will be prompted for only those signs, symptoms, and causes that distinguish the acute form. Those common to all forms of myocardial ischemia will be inherited from the parent class. In addition, the components will be checked to satisfy all the invariants inherited through the class hierarchy, in addition to any further constraints that may be specified for Acute Myocardial Ischemia.

Expected benefits and further requirements

To make practical use of the formal model, we had to develop a knowledge editor that essentially acted as a syntax-directed editor and type checker. For this

application, we felt that the sheer volume of information required was far too large to generate a paper copy of the specification of the domain knowledge. The editor "compiled" the knowledge directly into the internal representation (Prolog clauses) that was used by the final system. It would be quite feasible to produce a tool that could generate a hard copy of the domain model in the syntax described, although we have not yet tried this.

This approach has these advantages:

- Strong typing. This can be used to catch

***WE FAVOR DECOMPOSING
DSSs INTO SETS OF MODELS,
BUT WE MUST ALSO ACCEPT
THAT THE DEGREE OF RIGOR
AND FORMALITY WITH WHICH
THE MODELS ARE SPECIFIED
WILL VARY.***

various errors, from misspelling a specific symptom name for a pathology, to checks on the integrity of relationships between concepts (a disease may be a cause of a pathology, but not a symptom, for example).

- Checking for incompatible values. The predicates used to define invariants for classes act as integrity constraints on the database. For example, we had designated that the Signs and Contra-signs for a Pathology must be disjoint. Consequently, if Polyarthritis is defined as a subclass of Arthritis (itself a Pathology), we may not input that Multiple Joint Involvement is both a Sign and a Contra-sign of Polyarthritis. This is a simple and obvious mistake, but we can also catch and flag more subtle cases where an attempt is made to record a Sign for a Pathology as a contra-sign for a subclass of that Pathology.

These advantages also provide useful constraints on the generation of the inference rules that are actually used in a specific task. The OSM uses generic rules for "lifting" domain knowledge into rules that can

be used for a specific task. The following is a simplification of what is actually implemented, but it illustrates the principle.

In a task of diagnosis, we can use a generic rule or schema such as

If S is a Sign of P, and P is a Pathology, then "S \Rightarrow P"

to generate a diagnostic rule S \Rightarrow P. This asserted rule says that the observation of S supports the diagnosis P. For example, application of this schema might generate the specific instance:

Multiple Joint Involvement \Rightarrow
Polyarthritis

Without any integrity constraints on the database, one might also (mistakenly) generate an incompatible rule such as

Multiple Joint Involvement \Rightarrow
Not Polyarthritis

As we have seen, the possibility of generating such a pair of rules is blocked at an early stage.

The use of a small number of schemas to generate the inference rules makes it straightforward to ensure against rule circularities. The cases of redundant rules and unreachable rules can also be covered. However, the relative ease with which some of these cases can be covered may be due, at least in part, to the relatively shallow inference structures that result from the OSM architecture.

There are very strong similarities between the OSM and the four-layer conceptual model of KADS.³ The latter partitions the model of expertise into domain, inference, task, and strategy layers. (We have only considered the domain, inference, and task layers. Indeed, most of those we have spoken to who are involved with KADS have said that they have not found much use for the strategy layer.) Both the KADS and OSM teams have an interest in formal languages to add precision to what has, to date, been a relatively informal expression of the components of the model of expertise. This is being addressed with the development of a number of modeling languages for KADS. One of these, (ML)², has a similar motivation to our work in that it is based on an extension of first-order logic. However, the distinction is that our

approach uses a model-based specification language (an object-oriented variant of Z), whereas (ML)² draws its inspiration from algebraic specification languages. Domain knowledge is specified in (ML)² using order-sorted logic, together with a module system for structuring the specification. The KADS team has found that different formalisms are needed for specifying the inference and the task layers. They propose the use of metalogic and dynamic logic, respectively, for these layers. That is, (ML)² consists of three different formal constructions: order-sorted logic for the domain layer, metalogic for the inference layer, and dynamic logic for the task layer.³

The important distinction is that (ML)² concepts are represented by constants, whereas our concepts are represented by objects with properties and structure. However, the concepts of (ML)² can be collected into classes that form the sorts in the sort hierarchy, and these in turn can be grouped into theories that express properties of the sorts and concepts. Further work and case studies are needed to establish whether there are any fundamental reasons for preferring a model-based approach (as in the previous discussion) over an algebraic approach (as in (ML)²). We cannot offer as rigorous a specification of the task and inference layers as can be obtained using (ML)²; this probably explains our strong interest in that project.

The formalisms we have referred to are meant to illustrate possible approaches. Various techniques may be necessary (as exemplified by the three formalisms already proposed as components of (ML)²). In agreement with the KADS methodology, we have proposed distinct specifications of the domain knowledge and the inferences and tasks that may draw on that knowledge. But a fundamental contention of this article is that specification should be extended to a wider context to include the intended user and the environment in which the system is to be used. When we presented at the Eurovav workshop last summer, we received the comment that this asked more questions than it answered. Here we have to agree, but we still think they are worth asking. Indeed, work is underway to model system-user interaction³ and, of course, a great deal of work is being undertaken in the field of human-computer interaction, which will have a bearing on this aspect.

GIVEN THE CURRENT STATE OF medical knowledge, there are limits to the extent to which we can specify the behavior of a medical DSS; skilled clinicians may well differ in their diagnoses or preferred treatments for a given patient. However, this does not imply a criticism of the potential usefulness of DSSs. Although we would expect DSSs to perform reliably and at their best, we cannot expect them to perform better than a competent expert in the same domain. The most appropriate action in any given context will be subject to professional judgment and may be open

AN EXPLICIT STATEMENT OF THE PROPERTIES OF THE MODELS WILL HELP ASSESS THE DEGREE OF TRUST THAT CAN BE PLACED IN THE SYSTEM, AND THE CONTEXT IN WHICH IT CAN BE USED.

to dispute between professionals. Consequently, we cannot expect to guarantee that a DSS will be able to identify the "best" decision, merely that due care has been taken in considering the possible options. If a medical DSS is regarded as a service, this limitation is recognized in law.

A more rigorous specification phase can help to provide increased reliability of and confidence in the resulting DSS. In favoring a decomposition of the DSS into a set of models, we must also accept that there will be variations in the degree of rigor and formality with which the respective models are specified. Again, this need not be considered a weakness, as an explicit statement of the properties of the models used in building a DSS will help assess the degree of trust that can be placed in the system, and the context in which it can be used. An explicit representation of the models would also open the possibility of their independent evaluation. This would be a significant advance in evaluating DSSs for use in safety-critical applications (although it may, of course, conflict with requirements for the protection of intellectual property).

So, we do not yet know if we can formally specify a medical DSS, but we hope this discussion will help to identify the research directions that could be taken to achieve that goal.

Acknowledgments

Paul Krause is supported under the Science and Engineering Research Council project 1822: A Formal Basis for Decision Support Systems. We thank the anonymous reviewers for providing helpful comments and advice on an earlier draft of this article.

References

1. J. Fox et al., "Logic Engineering for Knowledge Engineering: Design and Implementation of the Oxford System of Medicine," *Artificial Intelligence in Medicine*, Vol. 2, No. 6, 1990, pp. 323-339.
2. J. Fox et al., "Using Predicate Logic to Integrate Qualitative Reasoning and Classical Decision Theory," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 20, Mar./Apr. 1990, pp. 347-357.
3. *Knowledge Acquisition*, special issue on the KADS approach to knowledge engineering, Vol. 4, No. 1, 1992.
4. M.C. Gemignani, "Some Legal Aspects of Expert Systems," *Expert Systems with Applications*, Vol. 2, No. 4, 1991, pp. 269-283.
5. T.J. Lydiard, "Overview of Current Practice and Research Initiatives for the Verification and Validation of KBS," *The Knowledge Eng. Rev.*, Vol. 7, No. 2, 1992, pp. 101-114.
6. J.M. Spivey, *The Z Notation. A Reference Manual*, Prentice Hall Int'l, London, 1988.
7. C.B. Jones, *Systematic Software Development Using VDM*, Prentice Hall Int'l, London, 1986.
8. K.L. Bellman, "The Modelling Issues Inherent in Testing and Evaluating Knowledge-Based Systems," *Expert Systems with Applications*, Vol. 1, No. 3, 1990, pp. 199-215.
9. A.J. Glowinski, E. Coiera, and M. O'Neil, "The Role of Domain Models in Maintaining Consistency of Large Medical Knowledge Bases," *AIME '91, Proc. of the Third Conf. on Artificial Intelligence in Medicine*, Springer-Verlag, Berlin, 1992, pp. 72-81.
10. S.A. Schuman and D.H. Pitt, "Object-Oriented Subsystem Specification," in *Program Specification and Transformation*, L. Meertens, ed., Elsevier North-Holland, New York, 1987, pp. 313-342.



Paul Krause is a research fellow at the Imperial Cancer Research Fund. His research interests include formal specification and development of software, logical models of reasoning, and reasoning under uncertainty. He is a graduate member of the Institute of Mathematics and Its Applications. He received his PhD in physics, as well as earlier degrees in mathematics and physics, from Exeter University.



John Fox is head of the Advanced Computation Laboratory and is responsible for research in artificial intelligence and knowledge engineering at the Imperial Cancer Research Fund. He is also a visiting professor at University College,

London, and founding editor of *The Knowledge Engineering Review*. His research interests include decision making, reasoning under uncertainty, and scientific theory formation. He received the first and PhD degrees in psychology from Durham and Cambridge universities, respectively.



Mike O'Neil is a research fellow at the National Health Service Centre for Coding and Classification, with a research interest in representing large medical thesauri to support problem solving. From 1986 to 1992, he was a clinical research fellow at the

Imperial Cancer Research Fund, where the research for this article was conducted. He is a member of the Royal College of Physicians. He received his MD from University College of London.



Andrzej Glowinski is a clinical research fellow in computer science at Manchester University. His research focuses on the development of advanced knowledge representation techniques for use in medicine, and the application of knowledge-based

methods for decision support in primary care. Between 1986 and 1992, when the research for this article was performed, he was a clinical research fellow at the Imperial Cancer Research Fund and worked on medical DSSs. He is a member of the British Standards Institute technical committee dealing with healthcare terminology, semantics, and knowledge bases. He received his MD from Oxford University in 1980, and then trained in general practice.

Readers can reach the authors in care of Paul Krause, Imperial Cancer Research Fund, 61 Lincoln's Inn Fields, London WC2A 3PX, United Kingdom; e-mail, pjk@acl.lif.icnet.uk

Knowledge-Based SYSTEMS

Quarterly

ISSN 0950-7051

ANNOUNCEMENT!.... ANNOUNCEMENT!.... ANNOUNCEMENT!....

Knowledge-Based Systems, the international, interdisciplinary and applications-oriented journal on knowledge-based systems is pleased to announce a new section, to appear in the journal: **Knowledge-Based Systems Letters**.


Knowledge-Based Systems Letters will provide a fast track for the publication of short contributions. These are intended to announce significant new results without extensive background information. They will be processed more quickly than papers and, most importantly, will not preclude the later publication of a full paper on the same topic.

Contributors are invited to submit **Letters** consisting of up to 2000 words and in the normal format given in our [Notes For Authors](#). Contributions should be sent to an appropriate member of the Advisory Editorial Board, who will arrange for review and make a speedy recommendation for publication to the General Editor. In this way, **Letters** are guaranteed the quickest route to publication in the very next issue possible.

For further information on **Knowledge-Based Systems Letters** and the submission of contributions, or to request a copy of our [Notes for Authors](#), please contact:

Karen Hemingway, **Knowledge-Based Systems**, Butterworth-Heinemann, Linacre House, Jordan Hill, Oxford OX2 8DP, UK.
Telephone: +44 (0)865 310366 Fax: +44 (0)865 310898

BUTTERWORTH
HEINEMANN

 A member of the Reed Elsevier group

