

# Disseminating medical knowledge: the *PROforma* approach

John Fox, Nicky Johns, Ali Rahmzadeh<sup>1</sup>  
Imperial Cancer Research Fund, Lincoln's Inn Fields,  
London WC2A 3PX,  
United Kingdom

## Abstract

Medical knowledge is traditionally disseminated via publication of documents and through participation in clinical practice. Information technology offers to extend both modes of dissemination, via electronic publishing and virtual reality training, for example. AI promises even more radical changes through the possibility of publishing clinical expertise in the form of expert systems, which assist patient care through active decision support and workflow management. *PROforma* is a knowledge representation language that is designed to support this new mode of dissemination. It is based on an intuitive model of the processes of care and a well-understood logical semantics. The paper provides a description of the language and associated software tools, and discusses its potential roles in, and implications for, medical knowledge publishing.

**Keywords:** medicine, knowledge representation, proformalisation, software engineering, formal methods, decision making, planning, uncertainty.

## 1. Introduction

Medical knowledge has traditionally been disseminated by means of apprenticeship and the private study of published material. Apprenticeship includes the observation of experienced clinicians at work and supervised participation in the care process. Study is via tutorial and reference texts, research journals and so on. These modes of dissemination are far from perfect. Apprenticeships are demanding of resources so there cannot be many apprentices. Furthermore, apprentices are trained to emulate their teachers, which is likely to perpetuate variability in medical practice. The publication process, on the other hand, fosters the dissemination of high quality and scientifically well grounded knowledge on a wide scale and the promulgation of good practices by the best specialists. It cannot guarantee, of course, that the student will fully absorb and properly apply the contents.

In recent years these two traditional methods have been augmented by new types of distribution media, such as video and the internet, and new forms of apprenticeship, such as telemedicine and virtual reality training. At the heart of all these technologies, however, are the same two basic modes of knowledge dissemination with their intrinsic limitations. Electronic recording of clinical practice does not correct weaknesses in the individual performance and, as we well know, internet publication does not guarantee

---

<sup>1</sup> Acknowledgements. Many people have contributed to the development of *PROforma*. We would particularly like to acknowledge doctors Andrzej Glowinski, Peter Johnson, Mike O'Neil, Jean-Louis Renaud-Salis and Robert Walton who have contributed their clinical experience and ideas; David Elsdon, Claude Gierl, Colin Gordon, Paul Ferguson, Saki Hajnal, Ian Herbert and Andrew Jackson-Smale made important contributions to earlier incarnations of the technology.

quality of content. And of course both modes are still limited by the abilities of a student to absorb and retain their lessons and then to apply them in new situations.

A third form of knowledge dissemination is now emerging which, for lack of an established term, we call *proformalisation*. This is a process of capturing clinical expertise in a form which can be directly applied by one agent (such as a computer) on behalf of another (such as an expert clinician or an authoritative organisation), hence the name<sup>2</sup>. The promise of computing in general, and AI in particular, is that it is becoming possible to publish a corpus of medical expertise which has been carefully validated and standardised, in a form that can be used to bring prompts, reminders and various kinds of advice to the point of care. This promises to help bring greater consistency and quality to routine clinical work, offering support for evidenced-based practice, tailored to the needs of the individual patient.

“Expert” and “decision support” systems currently offer the leading approaches to proformalisation of medical procedures. Early decision support systems were based on formal mathematical models from decision theory, operations research and the like. The current emphasis, however, is on “knowledge-based” systems which use symbolic rules and other relatively informal representations to capture clinical concepts and problem solving methods. The resulting knowledge representations can be very appealing. However, there are strong arguments that to fully achieve the potential of the new forms of knowledge publishing we need to be rather more formal than we are currently (e.g. Fox, 1993).

One argument draws on an analogy with a technical development that occurred about a thousand years ago which had vast repercussions for education, training, and professional practices that continue to the present day. This development was not in medicine but in music.

Around the turn of the millennium the repertoire of all singers and instrumentalists was disseminated entirely through live performance. There was no other way. The vagaries of human memory and preference naturally resulted in variation from one performance to another, with pieces forgotten, or added, or adapted as it pleased the artist. Consistency and no doubt quality were variable as a result. These shortcomings inspired one Guido d’Arezzo, a monastery choirmaster in Tuscany to develop the basic concepts of modern musical notation.

Musical notation, like other kinds of mathematical notation, is not only clear and precise, it also possesses an important property with deep implications. Musical notation has both a procedural interpretation (it can be played by people or machines) and a declarative aspect (it can be written, published, understood, discussed, edited etc.). Leading performers of the day saw the new notation as an attack upon their creativity, but its virtues were irresistible. It made possible a clear record of the composer’s intentions for the performers, and led to the very concept of music publishing and the whole of the modern music industry.

There have been many attempts to define knowledge representation languages that are capable of capturing general medical knowledge and clinical procedures. For example the “Arden syntax” is a Pascal-like language intended to provide a standard format in which medical knowledge can be specified and shared (Hripzak et al, 1994). However, the Arden syntax and other languages known to the author do not provide a strong underlying clinical performance model to guide the application developer, nor do they offer the declarative properties which are needed for effective and safe dissemination of expertise.

Proformalisation may offer an important new mode for medical knowledge dissemination. However, this potential will only be achieved satisfactorily if a balance is struck between the intuitive, but informal and imprecise representations offered by current expert systems technologies and a more formal and deeply understood representation of medical expertise. A number of efforts are under way to achieve improvements in specific areas, such as knowledge acquisition and reuse (e.g. Musen et al, 1992), standard interchange languages (e.g. Ohno-Machado et al, 1997), improved safety management (e.g. Hammond et al, 1994) and

---

<sup>2</sup> A contraction of proxy (“authorised to act for another”) and formalise (“give definite form to”).

stronger theoretical foundations for AI systems generally (e.g. Das et al, 1997a). This paper describes *PROforma*, a knowledge composition language and associated performance technology, which attempts to bring together the lessons for knowledge dissemination from such efforts within a framework that provides an expressive but precise formalism for medical expertise. It includes a more complete presentation of material introduced in Fox et al (1996).

## 2. Examples of proformalised expertise

Figure 1 shows a simple example of the kind of clinical application we have in mind. CAPSULE is a decision support system designed to advise on the prescribing of medications for common conditions in general practice. CAPSULE was designed by Dr. Robert Walton, a general practitioner. Its intended use is illustrated by the following scenario.

A patient visits his GP with long-standing mild osteoarthritis, for which he has been previously prescribed naproxen, but his symptoms are no longer fully controlled. After reviewing the history, current medications etc. (the relevant information is shown in the top half of the display), a British GP would normally consider the various medications he knows of and commonly uses, come to a decision whether or not to revise the dose or drug, and write out a prescription order for the pharmacist. If CAPSULE is used, however, the doctor presses a button requesting suggestions. In response the system examines the patient record, consults its comprehensive knowledge base of drugs and their uses, and displays a short-list of potentially suitable medications. The user can ask for an explanation of the pros and cons of any of the drugs on this list, an example of which is shown in the inset at bottom right. Here the user has asked for the arguments for and against naproxen, which the patient is currently taking though it is not at the top of the list of CAPSULE's recommendations. In this case CAPSULE reports that the drug is a generic treatment of choice for this condition according to the British National Formulary, and it appears from the notes that it has been effective in the past as well as suiting the patient because of lack of side-effects. However the patient has a history of asthma, and naproxen is contraindicated in this situation. Rather than simply increase the dose of naproxen, therefore, the GP accepts CAPSULE's suggestion that paracetamol be substituted.

### Figure 1 about here

In a systematic study with 42 GP volunteers, Walton et al (1997) found that this kind of decision support can help to make decisions more quickly and improve the quality of decision making quite dramatically, increasing the effectiveness of prescribing without increasing cost.

Figure 2 shows an example of a more complex guideline, for advising on the management of asthma in the accident and emergency department of a hospital, which has been proformalised using the present approach.

A young man arrives at an accident and emergency unit in the middle of an acute asthma attack in the early hours of the morning. Unfortunately no experienced staff are immediately available, so the junior doctor on call asks the department computer to locate a guideline for asthma management. It proposes one originally published by the *British Thoracic Society*, but which has been converted from the original published text so that the computer can "enact" the guideline, and assist the doctor through what is an unfamiliar procedure.

### Figure 2 about here

Figure 2 shows the guideline at a point early on in this enactment process. The first panel shows the guideline rendered as a set of tasks in a structured "task network" or workflow diagram. (Access to the

original guideline as a multimedia document and other published reference information is also available.) The second panel shows the point at which the doctor is completing the first task, “initial assessment”. The enactment process has provided the user with various reminders of what information to record and data entry dialogues, and the computer has suggested that the severity of the attack should be provisionally classified as “moderate” according to *BTS* criteria.

The *BTS* recommended procedure for managing moderate asthma in adults is now followed. This procedure consists of reassessing the patient after 30 minutes, possibly with suggestions for further medication, and again after 60 minutes. The system prompts for any relevant information at the appropriate time. As the tasks are enacted the graphical interface shows which tasks are in progress, which pending, which complete, and so on. The system also monitors the record of patient data for information that might indicate a misclassification of the patient, or any other adverse event. Such watchdog functions may even be linked to external monitoring devices.

### 3. The *PROforma* representation

The heart of the *PROforma*<sup>3</sup> approach is a language for representing medical procedures and decisions, and the medical knowledge and patient data that they use. It is a *formal specification language* in the sense used by software engineers; a formalism which helps to ensure clear, logical design and the verification of quality and safety. It is also a *knowledge representation language* in that it represents familiar clinical tasks, concepts and reasoning in a way which seems natural to medical professionals yet is sufficiently well-defined to be interpreted unambiguously by a computer.

*PROforma* can capture an indefinite range of clinical tasks, such as decision making, scheduling of actions over time, reminders for patient data collection, monitoring of adverse events, and so on. Furthermore, tasks may be grouped together to represent complex clinical procedures such as research protocols or routine guidelines. Once a medical procedure has been represented in the language it can be used in various ways. It can be reviewed by specialist clinicians and scientists to ensure that it captures “best clinical practice”, disseminated electronically as a source of up-to-date reference material, and enacted by a computer in order to assist medical staff to follow the recommended process.

#### 3.1. The *PROforma* model: process view

Figure 3a summarizes the process model which underlies the *PROforma* language. Each node of the “domino” represents information about a particular clinical situation, facts about a patient’s history, present activities or planned care. Each arrow represents an inference procedure that generates information of the type shown at its head from information of the type shown at the tail together with information in the patient record and/or general medical knowledge base.

**Figure 3 about here**

The domino model is a general framework within which to define domain-specific rules and other knowledge needed to instantiate the kinds of inference which are needed for a specific application. It is formalised in terms of classical predicate calculus augmented by certain non-classical (first-order) logics.

---

<sup>3</sup> The *PROforma* technology is being developed as part of the PROMPT project (PROtocols for Medical Procedures and Therapies) which is funded under the European Union’s Programme on Healthcare Telematics, 1996-1998.

The types of operation supported by PROforma are as follows

<i>Triggers</i>	The occurrence of clinical states, trends or other events may trigger the recognition of clinical problems which require solution.
<i>Problem solving</i>	Identify possible solutions to a problem. If the problem is to diagnose a patient then the candidate solutions will be alternative explanations of the patient's condition; if the problem is to treat a condition then the candidates may be alternative drugs or care plans (Huang et al, 1994). This step can be formalised by conventional deductive logic or abductive methods.
<i>Argumentation</i>	Argumentation is a general method for assessing the strengths and weakness of alternative solutions to a clinical problem. Argumentation has been formalised in the logic of argument, LA (Fox et al, 1993). This provides a qualitative basis for decision making, but can be augmented with a variety of quantitative calculi (Krause et al, 1995).
<i>Accept/Adopt</i>	Clinicians generally face two classes of decision; decisions about what to believe (e.g. diagnosis) and decisions about what to do (e.g. therapy). Both kinds of decision are informed by assessing the relative strengths of arguments for and against the alternatives, yielding a preference order on the set of alternatives. A simple preference order, however, may not be sufficient and other criteria for accepting hypotheses (e.g. parsimony) or adopting plans (e.g. relative safety) may be needed (Fox and Parsons, 1998).
<i>Scheduling</i>	Once a care plan has been adopted its "enactment" typically consists of a sequence of clinical actions carried out over time. PROforma supports a well-defined scheduling procedure in which tasks pass through a series of states, from pending to active to performed etc. Scheduling is formalised in the domino model by $L_{OT}$ , the Logic of Obligation and Time (Fox and Das, 1997b).
<i>Data acquisition</i>	The final types of inference in the model are concerned with the interpretation of clinical data. Data may be collected by manual entry (e.g. by forms) or by automatic means (e.g. from monitoring devices). These details are not presently part of the PROforma model. However, reasoning from raw clinical data to yield symbolic, temporal and other abstractions (such as normal/abnormal; rising/falling etc.) is supported.

A specific PROforma application may instantiate part or all of the domino model, and possibly draw upon a separate body of general medical knowledge.

For example, the CAPSULE system makes use of the decision making part of the domino (left-hand side as shown in Figure 3b) together with a knowledge base about drugs and their uses. Decision support is invoked manually in the current version, but it can be offered automatically; the trigger could be a provisional diagnosis being recorded in the patient record, for example. CAPSULE then generates a set of candidate medications from its general knowledge about drugs, and arguments pro and con each alternative are derived from specific information about the patient together with knowledge about cost, efficacy, side-effects, interactions and contraindications of drugs (Walton et al, 1997).

In the case of the BTS asthma guideline the domino model is used to automate the recommended care process using both the decision making and plan enactment parts of the domino model (left and right-hand

sides respectively, as shown in figure 3c). According to the guideline, the first clinical task on arrival of an asthma sufferer is to assess the severity of the patient's condition, for which there are four alternatives (mild, moderate, severe and life-threatening). The guideline recommends the collection of relevant data to permit the classification and proposes an appropriate decision if the *BTS* criteria are satisfied. As with CAPSULE the decision whether or not to accept the proposal is left to the clinical user<sup>4</sup>.

The level of severity determines the appropriate treatment routine. These routines consist of various clinical tasks carried out over about an hour, including the giving of medications, recording patient response, reviewing the patient's condition and so on. If the patient deteriorates during this process, a rescue or other action may be triggered, requiring the patient to be admitted into hospital, with further data collection and decision making and a more vigorous treatment regime initiated.

The *PROforma* specification language directly supports the concepts in the domino model (problems, arguments, decisions, plans, actions etc.) within an object-oriented framework, as set out in the next section.

### **3.2. The *PROforma* model: object-oriented view**

The aim is to achieve a balance between the clarity and precision of a formal specification language, and a more intuitive representation of clinical concepts and procedures. The domino model provides a good framework for defining *PROforma*'s formal semantics (Das et al, 1997a) but it is rather abstract. We have therefore developed a complementary view in which clinical procedures are conceived in a way that emphasises the clinical activities which the application is designed to support. These are naturally captured using the concepts of object-oriented programming..

In the object-oriented view of *PROforma*, an application such as a protocol or clinical guideline is modeled as a *plan* made up of one or more *tasks*. *PROforma* supports four basic classes of task (figure 4):

- A *decision* is a task that involves a choice of some kind, such as a choice of investigation, diagnosis or therapy. The *PROforma* specification of a decision task defines the decision options, relevant information and rules by which the pros and cons of different options can be derived.
- An *action* is a clinical procedure that is to be enacted outside the computer system, such as the administration of an injection.
- An *enquiry* is an action that yields information. The specification of an enquiry includes a description of the information required (e.g. the type, range and other properties of a clinical parameter) and a method for getting it (e.g. by creating a data entry form, by querying a local patient record or a remote laboratory database).
- Finally, a *plan* is a set of tasks, which are to be carried out to achieve a clinical objective, such as a therapeutic objective. Plans are composed of any number of *PROforma* tasks, including sub-plans, and are usually ordered to reflect logical, temporal or other constraints.

#### **Figure 4 about here**

Tasks are modelled in an *object-oriented style* in the usual sense that:

- (a) Any specific clinical task is seen as an instance of some more general class of tasks. Each class is located in a task ontology, as shown in figure 4 and is eventually a specialisation of a *root task*, (at the top of the figure). Any class can be further specialised into particular sub-types. For example, decisions can be specialised into diagnosis decisions and therapy decision, plans may be specialised into chemotherapy protocols and disease management guidelines.

---

<sup>4</sup> It is possible to specify that *PROforma* decisions can be taken automatically by a computer but we generally specify that decisions with non-trivial consequences are authorised by a clinically trained user.

- (b) Every task inherits part of its specification from the classes above it in the hierarchy, expressed as a set of attributes. A task is distinguished from its parents by the possession of additional attributes, and distinguished from its siblings by different values for their common attributes.
- (c) When tasks are enacted, the communication of information between tasks is achieved by passing messages between encapsulated task objects, rather than explicit procedure calls or other mechanisms which require access to the internal definition of a task.

The following tables summarise the attributes of the four task classes in the current *PROforma* ontology. In each table the first column is the attribute, the second shows an example value(s) and the final column gives a brief explanation.<sup>5</sup> We first describe the attributes of the root class which are common to all four task types, and then summarise the attributes which distinguish the four types.

## Root task

All tasks have the attributes shown in table 1. Where the attribute has an exclamation mark next to it, it is mandatory to include one or more values for the attribute.

### Table 1 about here

An application may not require all these properties. To avoid *PROforma* being a “sledgehammer to crack a nut”, most of them are optional. They are provided for those applications which require greater flexibility, and with the long term aim in mind that designers should be able to smoothly upgrade simple applications to more sophisticated ones as their experience grows or needs change.

## Plans

Plans define sets of tasks which may be required in a clinical procedure, such as a protocol, and may define logical and temporal constraints on their enactment, together with any circumstances in which the plan should be aborted or terminated (table 2).

### Table 2 about here

The only attribute of a plan that *must* be specified is its components. Consequently the simplest form of plan is just a *set* of tasks which must all be carried out but not in any particular order. A more typical plan may be used to capture a clinical flow diagram in which there are constraints on the ordering of tasks together with some conditional branching implemented by means of decisions. As has been shown in other work, such features are sufficient for defining many clinical routines. However, *PROforma* provides more expressive power than the flowchart notation, in order to permit the author to specify temporal constraints, abort conditions and so forth, which will be necessary for complex applications such as cancer chemotherapy protocols or clinical trial protocols (Shortliffe, 1986, Hammond et al, 1994).

---

<sup>5</sup> Note that this description (designated version 1.5) is significantly more complete than earlier accounts (e.g. Fox et al, 1996) since we are able to draw on a wider range of applications experience. However, although the current language definition appears to be quite stable, it should still be regarded as provisional. Following wider discussions with the community it is intended to publish a formal definition of the syntax and operational semantics of the language in the near future.

## Decisions

*PROforma* offers more expressive power than flowcharts and similar notations because it incorporates a well-defined model of decision making under uncertainty (Fox et al, 1990; Huang et al, 1994; Krause et al, 1995). Decisions are defined in terms of a set of options and the rules by which *arguments* for choosing alternative options may be generated.

### Table 3 about here

Experience to date suggests that the generic decision structure summarised in table 3 is sufficient to implement many types of clinical decision, including diagnosis, therapy selection, prescribing, referral and risk assessment decisions.

*PROforma* is designed throughout to permit flexibility. In the case of decisions, we anticipate that some decisions will involve significant uncertainty while others are categorical. The ability to make decisions under uncertainty is provided by means of the argumentation mechanism, but if this is not needed then the argumentation attribute can be ignored and a decision can be taken solely by means of the commitment mechanism.

## Actions

An action is a task that a *PROforma* application can request to be enacted by an external agent (table 4). The agent may be a clinical user (e.g. a nurse is requested to give an injection) or another software agent or hardware device (e.g. an infusion pump).

### Table 4 about here

An action can be used to represent a wide variety of operations within a plan, from administering medication to printing a leaflet, to ordering a test or carrying out a surgical procedure. These are all just specialisations of the action class.

From *PROforma*'s perspective, actions are atomic (they do not decompose into subtasks) however complex the external process. Even an action such as "heart bypass surgery" will be viewed as atomic: it can be requested, perhaps with confirmation that it has been completed by the external agent, but that is all.

## Enquiries

An enquiry is a task that is intended to return information (table 5). This may be clinical, administrative or other information and may involve a simple process, such as displaying a data entry form, or a complex one as in extracting information from an image or capturing physiological signals from patient monitoring equipment. As with an action an enquiry method can be implemented with an attached procedure that is encapsulated within the enquiry task. Like actions, enquiries are atomic. (In fact, one might argue that both classes are just slightly different specialisations of a more generic "interface task". However, this would add complexity to the task ontology without any obvious benefit.)

### Table 5 about here

This concludes the summary of the *PROforma* language structure. Overall, *PROforma* can be classified as a logic language with extra-logical features including object-oriented features. Interpreters or compilers that permit the application to be tested and embedded in legacy applications may be implemented in a variety of technologies. One example, implemented in Prolog, is described later. Since *PROforma* has a well-structured

syntax, defined in a BNF-equivalent normal form, applications can be developed using ordinary text editors and compilers. However, in order to simplify the composition and testing of an application, a graphical authoring environment has been constructed which builds on the object-oriented features in order to insulate the application developer from many details.

## 4. The *PROforma* composition method

Development of a computerised care guideline or clinical protocol using the *PROforma* composition system is a two-step process. In step 1, a high level description of the guideline is developed using a graphical design package. In step 2, the graphical outline is fleshed out into a detailed process description.

### ***Step 1: Graphical Representation of the Task Network***

Starting from available knowledge sources, such as authoritative published guidelines, an application designer typically lays out a network of objects representing the decisions, actions and other tasks which are required to implement the guideline. Figure 5 shows a screen from the *PROforma* protocol editor during the creation of a guideline for the management of dyspepsia in primary care.

Figure 5 about here

### ***Step 2: Populating Task Templates***

For each task the designer is provided with a set of attribute frames, or templates, with which to enter the terms which are the values of the attributes. Templates guide the application designer in structuring the task specification required by the application. Every task has a standard set of templates to enter values for the standard attributes inherited from the root task, as illustrated in figure 6:

Figure 6 about here

The *PROforma* composition software provides a variety of tools to support the development process. For example, the templates provide syntax checking and data dictionary management, and the complete specification can be checked for consistency and completeness in a number of ways, drawing on the task model. If a *PROforma* composition passes all these checks it can be passed to the *PROforma* enactment engine to check that it is performed as intended.

## 5. The *PROforma* enactment engine

Figure 7 is a view of the test environment for checking correct enactment of an application, showing all the active tasks about half way through execution of the dyspepsia guideline. The window is showing the currently active tasks, organised as a hierarchy showing the component structure of the application with each component labeled using the appropriate task icon. For each task class a suitable viewer is provided to examine details of the current enactment state.

The task which is currently being enacted is a decision whether to prescribe cimetidine or ranitidine (H2 blockers) for which two items of information are required at this point. The enactment engine has generated a data entry form; this is shown at the bottom of the window.

The test interface provides access to all information about the dynamic state of an application through the various task viewers. It can be used, as here, as a stand-alone application. However the enactment software is designed to be embedded in “legacy” applications such as practice workstations or hospital information systems, in order to add value to existing systems by bringing care guidelines and decision support

functions to the point of care. The tester would not generally be expected to provide an appropriate clinical user-interface in these circumstances. Rather, we would expect application developers to build an interface which is suitable for the particular clinical setting and communicates with the engine via a standard API; the graphical interface of the asthma guideline above illustrates a specialised GUI linked to generic enactment software.

**Figure 7 about here**

The *PROforma* engine can be used stand-alone but we would normally expect it to be used in concert with a separate electronic medical record, and a database for storing application libraries (such as libraries of standard guidelines and research protocols) and a general medical knowledge base.

## 6. Discussion and implications

The *PROforma* language appears to satisfy a number of the objectives set out in the introduction. Most of the language can be understood in terms of the formal semantics of the domino model described by Das et al (1977b) yet the basic constructs of the process and object models seem natural and intuitive. It offers a declarative interchange format for describing clinical guidelines together with a knowledge acquisition methodology and tool set to simplify the composition and formal verification of applications. It also appears to be expressive, in that we have been able to implement demonstrator applications for a wide range of decisions, protocols and guidelines in a number of medical specialties.

Tools of this general type can be used to assist authoritative clinical and research organisations oversee the preparation of clinical guidelines and protocols and distribute them in a way that is complementary to conventional publication. An analogy with HTML and the World Wide Web is appealing: users with software which can enact a proformalised specification<sup>6</sup> will be able to download clinical guidelines from a “content provider’s” web site, offering an efficient and potentially low cost way of disseminating new knowledge and practices.

### 6.1. Proformalisation and evidence-based medicine

The proformalisation concept seems highly relevant to the so-called movement for evidence-based medicine. Countless individuals and organisations are now participating in an international effort to improve the consistency and quality of care by carrying out systematic reviews of research on alternative therapies and other medical procedures, and then distilling the results as scientifically validated clinical guidelines. This effort is seen by many as a revolutionary movement that offers a more objective and science-based approach to developing clinical practice than has been the norm in the past.

The current infrastructure of EBM seems to have a weakness, however; the results of these efforts are only published in the traditional way, on paper. This must confront the reality that busy doctors and healthcare professionals often have little opportunity to read, absorb, and subsequently apply the contents of such publications.

Proformalisation offers a new relationship between the composers of evidence-based guidelines and the clinicians whose performance they wish to influence. Figure 8 illustrates this relationship.

**Figure 8 about here**

---

<sup>6</sup> We are using the proformalisation concept in a general way here, since we expect that alternatives to *PROforma* will emerge but the basic composition-performance model will still hold.

Proformalisation can start at the point where clinical research usually ends, with the publication of a validated guideline on paper (phase 1 in figure 8). At this point, the application developer can begin a process of translating the validated evidence-based guidelines into a form that can be enacted on a computer (phase 2 in figure 8).

It is now possible to test the logic of the computerised guideline against realistic data (phase 3), though at this early stage this might not be in a clinical setting. The CAPSULE trial, for example, was carried out with a standard set of “paper patients”. Phase 3 is aimed at initial proof of concept and progressive refinement of the medical content to meet suitable quality criteria. Indeed, it has been known for a long time that developing a computerised form of clinical protocol can reveal incompletenesses or inconsistencies in the original text.

Phase 4 represents the first phase of clinical testing, in which the new guideline is evaluated in controlled clinical conditions using standard randomised trials techniques. If the outcome data from the trial warrant it, then the system could be *published* for use in routine clinical settings (phase 5). Use of the internet as a publishing medium is attractive because of the simplicity and speed of this process.

Proformalisation offers a final important bonus. If we are using electronic decision support systems, it should be straightforward to store all clinical information about patients, including the decisions that were taken and the reasons for those decisions, as well as which actions were carried out and when. The resulting database is a rich source of information for clinical audit and local policy assessment. If outcome data are also recorded this can be automatically fed back to the research community that developed the guideline in order to quantify its benefits and inform revisions.

## **6.2. Improving safety as well as quality of care**

Issues of quality and safety are paramount in some industries, such as the aerospace and nuclear industries. The implications for software developers in these fields are enormous, forcing the development and use of many systematic techniques such as hazard analysis, formal specification and verification, and software validation.

Expert systems have enormous potential to help improve consistency and quality of patient care, but if present approaches and trends continue they could well have effects other than those that are anticipated. Consider the following remark by Nancy Leveson:

“A negative effect was achieved when a major airline, known for having one of the best maintenance programs in the industry, introduced an expert system to aid their maintenance staff. The quality of maintenance fell: The staff began to depend on the computer decision making and stopped taking responsibility and making their own decisions. When the software was changed to provide only information and only when requested, quality again rose.” (*Safeware*, 1995, p 449)

Without wishing to exaggerate the similarities between clinical medicine and the processes of aircraft maintenance, the message is clear: like every other medical technology, decision support systems have the potential to improve the efficacy and/or safety of care, but also the potential to make things worse.

There is a growing consensus that, as with other new clinical technologies, decision support and guideline systems must be empirically validated in properly controlled clinical trials (Heathfield and Wyatt, 1994). What is missing, however, is an emphasis on the development of formal design theories and the incorporation of explicit safety mechanisms. Safety means safety in depth. Aerospace companies don't just test-fly aircraft to make sure they work as expected, they put a great deal of research and development effort into developing formal theories of airframe design, airflow and turbulence, stress distribution, software structuring, and so on. Without comparable developments in medical AI, providing a basis for establishing

and policing clear quality and safety standards, we run a risk of creating clinical disasters that bring the technologies into disrepute (Fox, 1993).

There is a long-established orthodoxy within the expert system community that expert knowledge is intrinsically informal (*heuristic*) and expert systems are consequently not amenable to the use of formal design techniques. This is in contrast to assumptions increasingly made in the aerospace, nuclear and other safety-critical industries. As a formal, declarative language, *PROforma* has some attractive features for automatic verification of applications. While we are incorporating basic checks for consistency, completeness etc. into our *PROforma* authoring tools, however, we share Leveson's view that absolute confidence in the reliability of software will continue to be a distant goal:

“Most standards and technical approaches to safety involve just ‘getting the software right’ or attempting to increase software reliability to ultrahigh levels. Although in the future it may be possible to construct perfect software, the current reality is that we cannot accomplish this goal for anything but the simplest systems. Moreover, even if the software had no errors, there would be no way to know this with high confidence.” N Leveson, *Safeware*, 1995, ix

We are also therefore addressing safety issues in a somewhat different manner, by incorporating features in the language which will permit applications to reason explicitly about the potential clinical consequences of decisions, plans and actions. There is considerable scope for such concepts (Fox 1993; Hammond et al, 1994; Fox and Das, forthcoming).

### **6.3. Standards for knowledge representation and interchange**

As in other fields of knowledge engineering it is widely recognized in medical AI that having a standard *interchange language* providing a common terminology and conceptual framework would have many advantages. Some of these are: the potential for reuse of knowledge modules and interoperability of applications; independence of applications from proprietary platforms; and the establishment of a market for experienced professionals with portable design and implementation skills.

There also seem to be advantages for industry, in that companies may choose to build software products, such as enactment engines for standard representation languages, just as they develop compilers for conventional programming languages, such as C, LISP or Prolog. This would permit competition for sales in the usual way based on price, performance, quality, reliability, safety features, and so forth. Perhaps more important it would permit companies to develop innovative products, such as advanced electronic patient records, ward management systems, interdisciplinary shared care packages and so on. This would permit them to find new ways of exploiting the medical knowledge published by content providers. They could innovate in this way while preserving the benefits of interoperability and reusability.

The development of standards for representing guidelines is obviously intimately linked with the development of standards for medical terminology and coding (e.g. Read Codes, SNOMED, UMLS) though traditionally they have been studied separately. In common with most other offerings in this area, *PROforma* does not commit to any particular terminologies, ontologies etc. The history of terminology research has until recently been fairly independent of AI, and it is not clear which of the current systems, if any, is most appropriate for the purposes described here. It is quite possible to build *PROforma* applications without adopting a global solution, but if the full power of proformalisation is to become available then standardization of medical terminology and knowledge representation are even more important and urgent than is already recognised.

## **References**

Cimino J J, Socratous S A, Clayton P D “Automated guidelines implemented via the world wide web”

- J. Am. Med. Informatics Assoc.* 2 (suppl), 223-227, 1995.
- Das S K, Fox J, Elsdon D, Hammond P "A flexible architecture for autonomous agents" *J. Experimental and theoretical Artificial Intelligence*, 9, 407-440, 1997a.
- Das S, Fox J and Krause P "A unified framework for hypothetical and practical reasoning (1): Theoretical Foundations" in D Gabbay and H J Ohlbach (eds) *Practical Reasoning*, Berlin: Springer, 1997b.
- Fox J "On the soundness and safety of expert systems ", *Artificial Intelligence in Medicine*, 5, 159-179, 1993.
- Fox J, Clark D, Glowinski A and O'Neil M "Using predicate logic to integrate qualitative reasoning and classical decision theory" *IEEE Trans. on Systems, Man and Cybernetics*, 20 (2), 347-357.
- Fox J and Das S K A *Duty of Care: Artificial Intelligence in Safety-Critical Applications* (forthcoming).
- Fox J, Krause P J, Elvang-Goransson M "Argumentation as a general framework for uncertain reasoning" *Proc. 9<sup>th</sup> Conference on Uncertainty in Artificial Intelligence*, Morgan-Kaufman, 1993.
- Fox J, Johns N, Rahmzadeh A, Thomson R "PROforma: a general technology for clinical decision support systems" *Computer Methods and Programs in Biomedicine* 54, 59-67, 1997
- Fox J and Parsons S "Arguments about values and actions" *Proceedings of Stanford Spring Symposium on Qualitative Decision Theory*, AAAI Press, 1997
- Hammond P, Harris A L, Das S K and Wyatt J "Safety and decision support in oncology" *Meth. Inform. Med.*, 33 (4), 371-381, 1994.
- Heathfield H and Wyatt J "Editorial commentary" *Methods of Information in Medicine*, 1994.
- Hripscak G, Ludemann P, Pruor T A, Wigertz O B, Clayton P D "Rationale for the Arden Syntax" *Computers in Biomedical Research*, 27, 291-324, 1994.
- Huang J, Fox J, Gordon C and Jackson-Smale A "Symbolic decision support in medical care" *Artificial Intelligence in Medicine*, 5, 415-430, 1993.
- Krause P J, Fox J, M O'Neill and A Glowinski, "Can we formally specify a medical expert system?" *IEEE Expert*, 56-61, 1994.
- Krause P, Ambler S and Fox J "A logic of argumentation for reasoning under uncertainty" *Computational Intelligence*, 11(1), 113-131, 1995.
- Musen M A, Tu S W, Das A K, Shahar Y "EON: a component based approach to automation of protocol-directed therapy" 3(6), 367-388, 1996.
- Musen M A "Dimensions of knowledge sharing and reuse" *Computers in Biomedical Research*, 25, 435-467, 1992.
- Musen M A, Tu S W, Das A K and Shahar Y 'A component based architecture for automation of Protocol-Directed Therapy', in Barahona P, Stefanelli M and Wyatt J (eds) *Proc. AIME95*, Berlin: Springer, 1995.
- Ohno-Machado L, Gennari J H, Murphy S, Jain NL, Tu S W, Oliver D E, Pattison-Gordon E, Greenes R A, Shortliffe E H, Barnett G Octo "The Guideline Interchange Format: A model for sharing guidelines" Technical Report, SMI-97-0668, Section on Medical Informatics, Stanford University, 1997.
- Shortliffe E H "Update on ONCOCIN: a chemotherapy advisor for clinical oncology" *Medical Informatics*, 11 (1), 19-21, 1986.
- Walton et al, "Effects of decision support on prescribing in general practice". *British Medical Journal*, 26<sup>th</sup> September, 1997.

## FIGURES

The screenshot displays the CAPSULE system interface for a patient named Mr X, aged 43. The interface is divided into several sections:

- MAIN PROBLEM:** mild osteoarthritis
- ASSOCIATED PROBLEMS:** chronic airways obstruction
- PAST HISTORY:** asthma, hypercholesterolaemia, impetigo, osteoarthritis, varicose eczema
- CURRENT DRUGS:** NAPROXEN 250mg tabs b.d. 2/52
- PATIENT PREFERENCES:** NAPROXEN 250mg tabs b.d. 2/52
- SOCIAL HISTORY:** (empty field)

Below these sections, a list of suggestions is provided:

Please select from suggestions:

- NAPROXEN 250mg tabs b.d. 2/52
- PARACETAMOL 500mg tabs two q.d.s. 2/52
- NAPROXEN 250mg tabs b.d. 2/52 (highlighted)
- IBUPROFEN 200mg tabs q.d.s. 2/52
- DICLOFENAC 25mg e/c tabs b.d. 2/52

Buttons for "Reasons behind suggestion" and "Alphabetical drug list" are visible.

Optional modifications: Dose: [ ] frequency: [ ] for: [ ] da

A pop-up window titled "This drug is suggested because :" provides the following reasons:

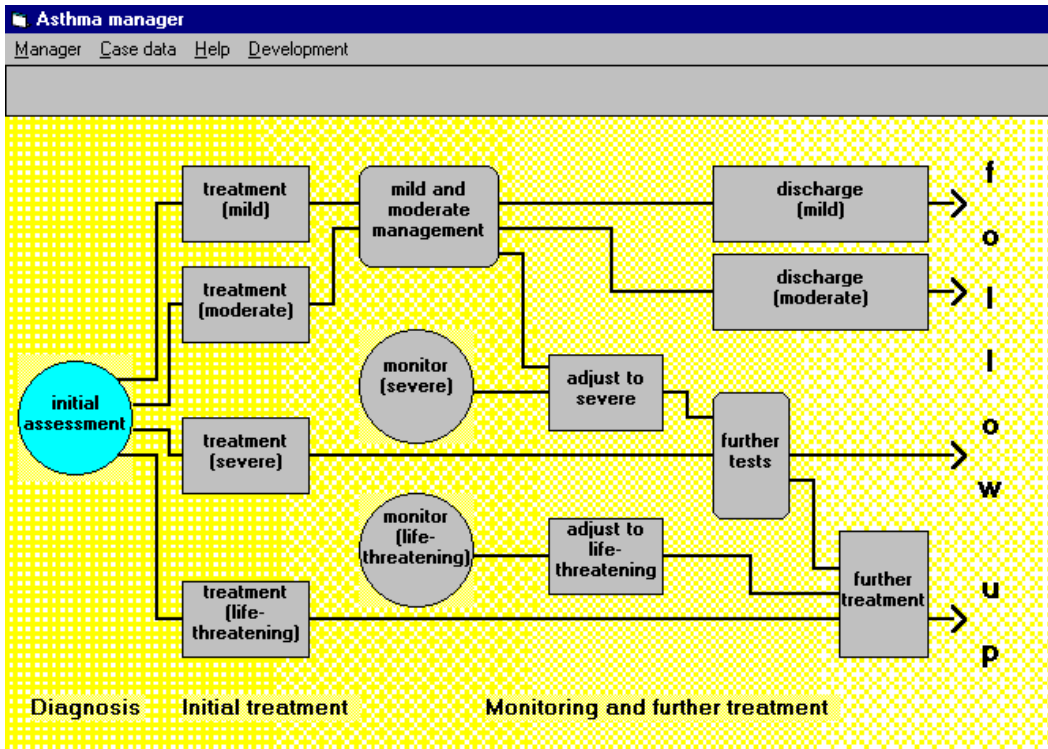
- \* It is a generic drug.
- \* It is BNF preferred treatment.
- \* It was effective in the past.
- \* The patient showed a preference for it.

However:

- \* It should be used with caution because of chronic airways obstruction.

An "OK" button is located at the bottom of the pop-up window.

Figure 1, the CAPSULE system being used to suggest medications for a patient with osteoarthritis. A demonstration of CAPSULE is available on the web at [www.infermed.com](http://www.infermed.com).



**Asthma manager**  
 Manager Case data Help Development

**Decision**

**Patient data**

PEFR	<input checked="" type="checkbox"/>
respiration	<input type="checkbox"/>
respiratory effort	<input type="checkbox"/>
chest	<input type="checkbox"/>
pulse	<input type="checkbox"/>
blood pressure	<input type="checkbox"/>
speech	<input type="checkbox"/>
pallor	<input type="checkbox"/>
demeanour	<input type="checkbox"/>
end	<input type="checkbox"/>
cor	<input type="checkbox"/>

**Possible condition(s)**

mild	<input type="checkbox"/>
moderate	<input checked="" type="checkbox"/>
severe	<input type="checkbox"/>
life-threatening	<input type="checkbox"/>

**initial assessment**

**Peak expiratory flow rate, initial value**

700  
600  
500  
400  
300  
200

PEF: 405

Peak expiratory flow in normal adults

**PEF Rate calculations:**  
 If Best PEF is not known, enter demographic details.

Best PEF:  l/min

Sex:  Female  Male

Age:

Height:  cm

**Calculate**

Diagnosis Initial treatment

Figure 2: Proformalised care guideline for the management of acute asthma in a hospital accident and emergency unit. The first panel shows an overview of the network of tasks required by the guideline. The second panel shows the user display during the enactment of the first task of the guideline assessment of the severity of the patient's asthma attack.

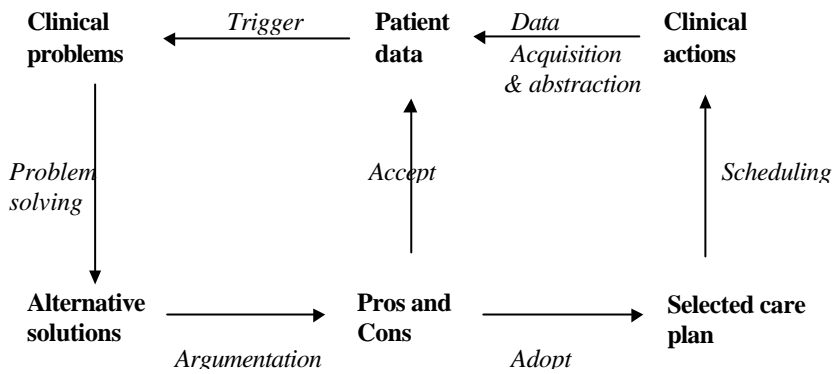


Figure 3 (a) : the “domino” clinical process mode.

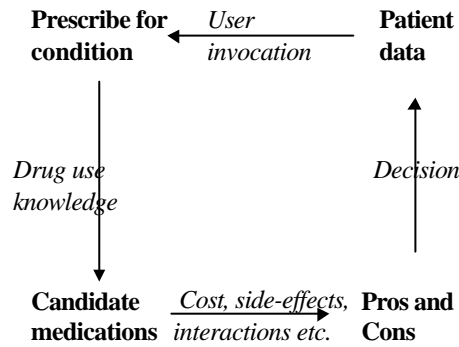


Figure 3(b): CAPSULE mechanisms as an instantiation of the decision making part (LHS) of the domino model

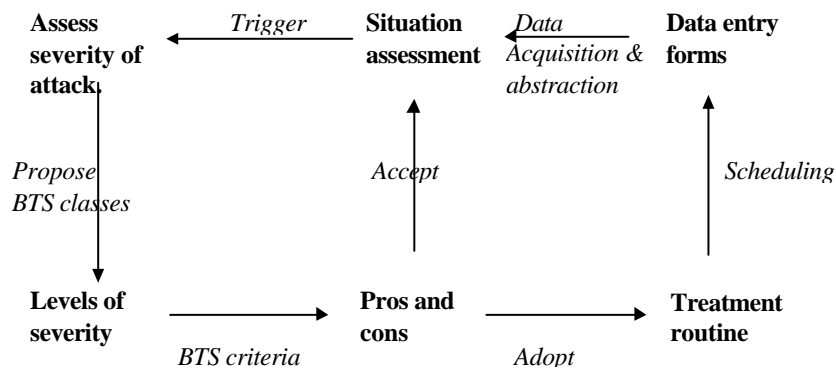
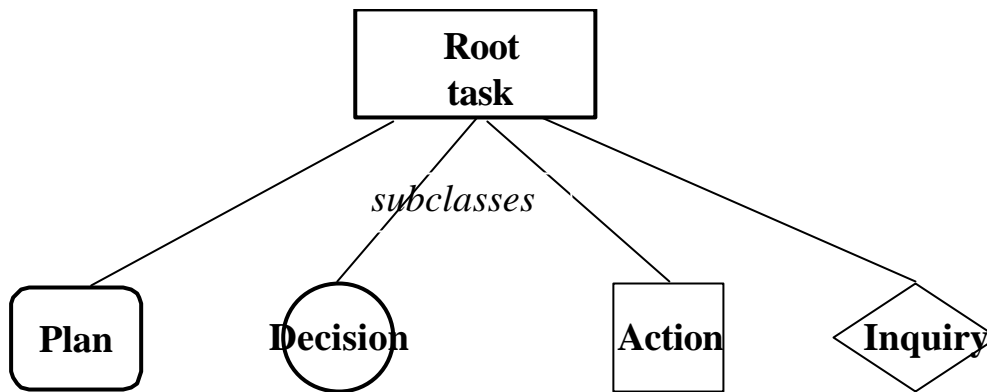
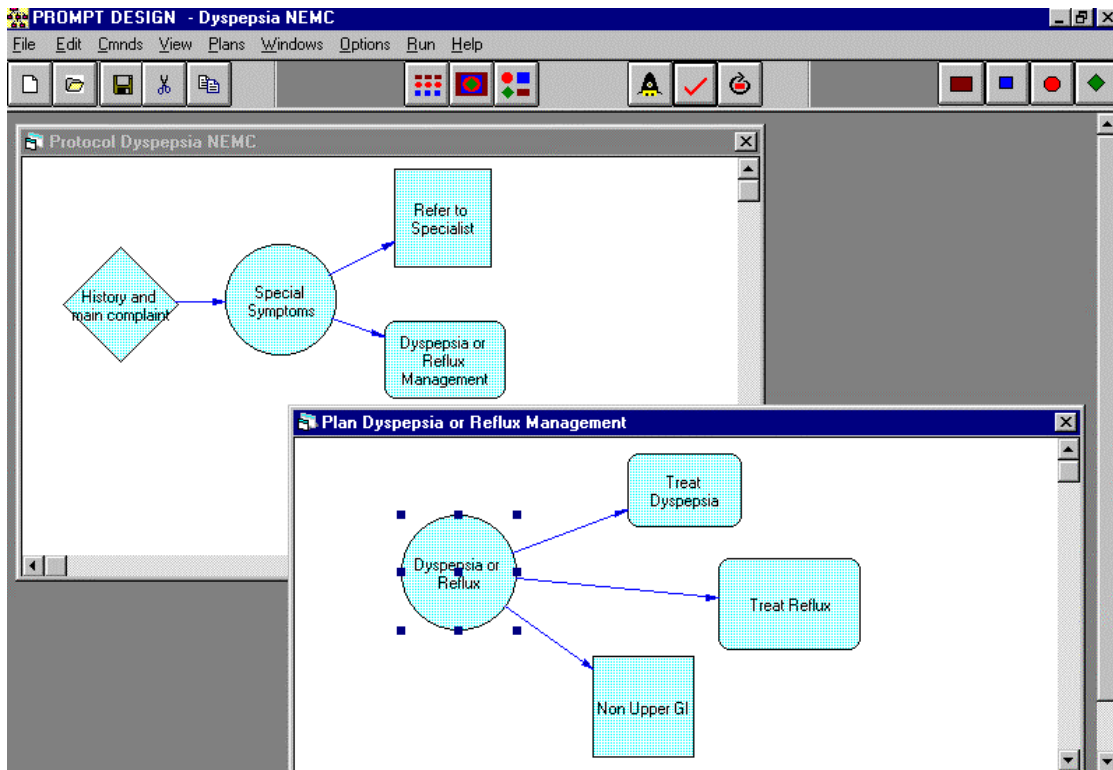


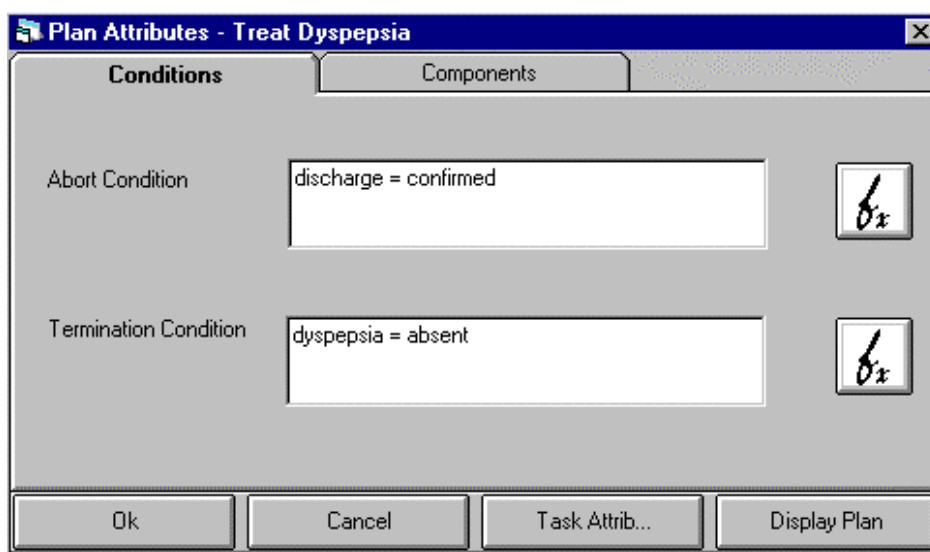
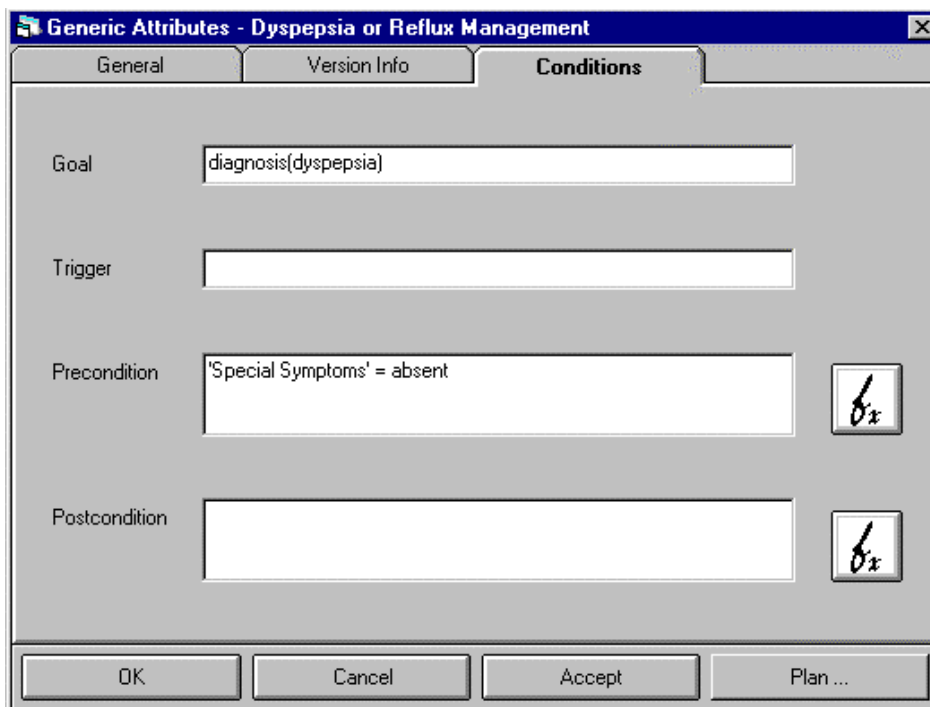
Figure 3(c): British Thoracic Society guideline for management of acute asthma in adults as an instantiation of the full domino model.



**Figure 4: The PROforma task ontology: decisions, actions and inquiries are “atomic” tasks, while plans are compound. Plans can consist of any number of atomic tasks, and subplans. All tasks have a set of common attributes (e.g. goals and preconditions) but each subtype has its own distinguishing attributes (see text).**



**Figure 5:** The PROforma editor provides graphical tools for laying out protocols in terms of decisions (circles); plans (round rectangles); actions (square rectangles) and inquiries (diamonds). Arrows are scheduling constraints. The rear window shows component tasks in a guideline for the management of dyspepsia in primary care; the front shows the decomposition for the subplan “dyspepsia or reflux management”.



**Figure 6: Examples of templates for populating PROforma tasks. The first shows the template for defining generic conditions of all tasks (which are inherited from the root task) such as the goal or preconditions of the task. Note that the goal example consists of a task type and a clinical focus, following Huang et al, 1994. The second example shows the attributes “Abort condition” and “Termination condition” which are specific to a PROforma plan. The function buttons are used to invoke a special composition tool for defining numerical and other functions which are to be incorporated in a condition.**

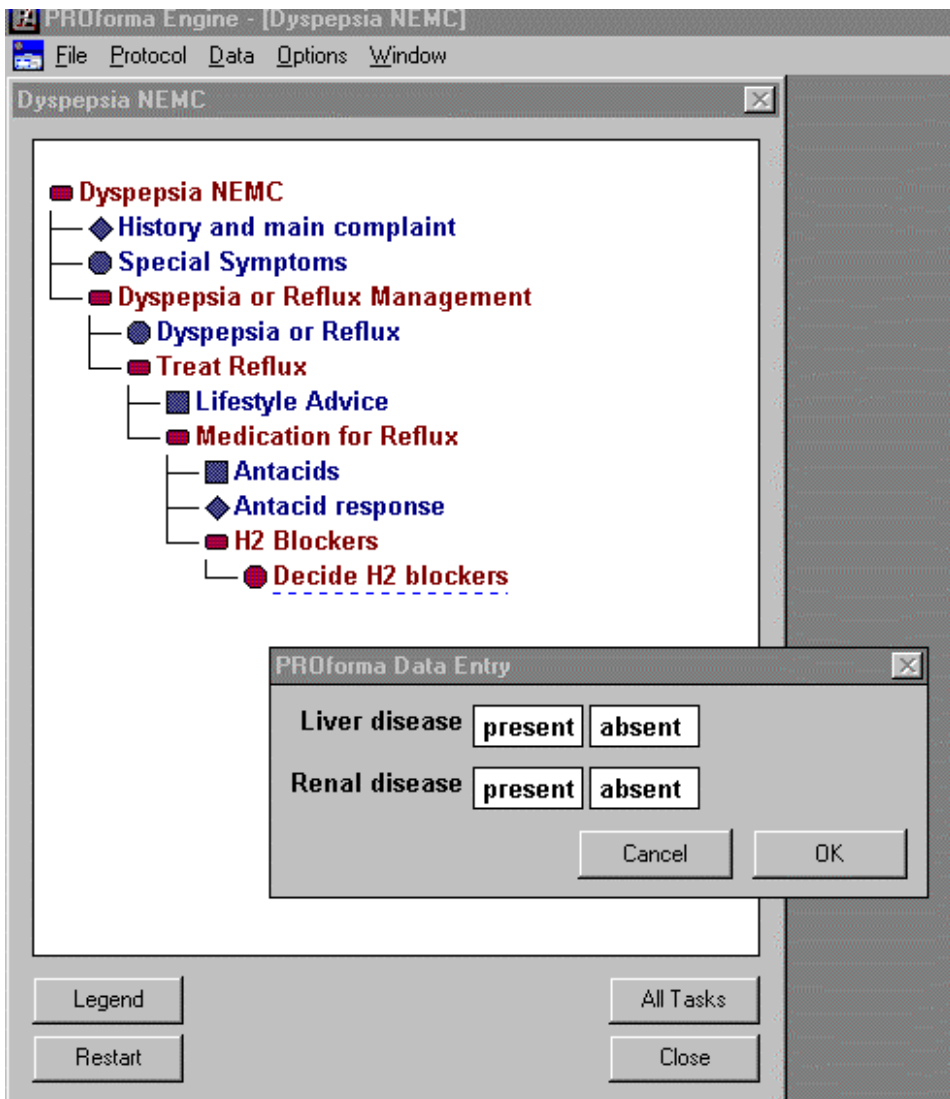


Figure 7: View of the PROforma enactment test interface

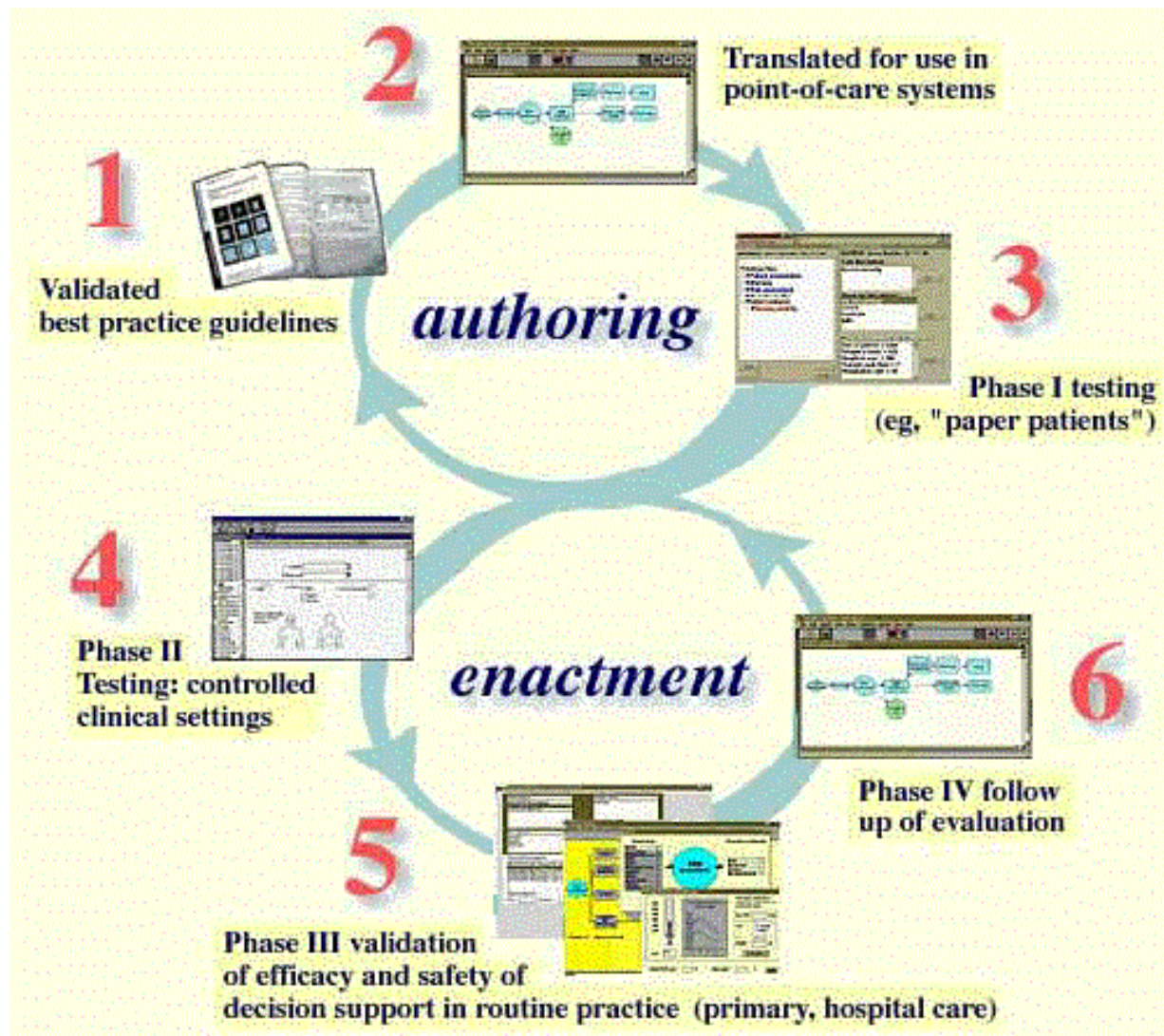


Figure 8: Synergy between decision support technology and evidence-based medicine. Following publication of a systematic review and meta-analysis of clinical trials the validated guidelines can be proformalised, tested, validated in a further trial and then released for routine use. Data acquired during such use can be collected and analysed (e.g. with data mining and statistical techniques) to provide input into development of the next generation of systematic review.

<b>Attribute</b>	<b>Example value</b>	<b>Explanation</b>
Name !	admit_1	Distinct identifier for the task
Caption !	Admit patient	Short informative text description for use where the name is to be pretty-printed.
Description	Admit into hospital if asthma attack is assessed as life-threatening	Documentation text, stored as a simple text string or accessed indirectly via a suitable application program (e.g. for documenting in hypertext or multimedia).
Goal	achieve(normal_respiration)	Optional term representing the clinical objective of the task. If a matching term is asserted into the patient record then the goal is deemed to have been achieved and enactment of the task will halt. If a task has no goal then enactment will be determined only by the task's scheduling constraints. If the goal condition already holds when a task is considered for enactment it will be skipped.
Precondition	risk_level = severe	This (optional) boolean condition must be true before the task may be enacted, even if all the task's scheduling constraints are satisfied. The value may be a set of if...then... rules in order to refine the preconditions.
Postcondition	patient_state = exhausted	Situational descriptions which will be assumed to be true at completion of the task. The value may also be a set of if...then... rules to refine the postconditions.
Trigger condition	1. notified(peak_flow < 30), 2. peak_flow < 30	A trigger may be a temporal event of some kind (1), or a boolean patient state description (2). Whenever the trigger condition becomes satisfied the task will be considered for enactment, but will only be enacted if preconditions also hold. Temporal events can occur any number of times and the task is considered for enactment on each occasion.
Cycles	cycle(Integer, Interval) cycle(until(State), Interval) cycle(while(State), Interval)	PROforma provides a cycle description sub-language to define conditions and constraints under which a task will be repeated. If a cyclical task is a component of a plan which is aborted or whose goal is achieved then cycling is terminated.

**Table 1: Attributes of the root task. These are common to all PROforma tasks, though values will typically be different across instances.**

Components !	history, diagnosis, therapy, follow-up follow-up	A set of task names which compose the plan.
Optional tasks		All tasks in a plan are assumed to be mandatory unless indicated as optional (e.g. to permit a user to skip the task).
Autonomous tasks	follow-up	Enactment of all tasks in a plan is assumed to require authorisation unless it is set autonomous, or the parent plan is autonomous
Scheduling constraints on tasks	history before diagnosis	A set of qualitative ordering conditions on tasks
Temporal constraints on tasks	followup after therapy at ten weeks	PROforma provides a temporal sub-language for including in the conditions of rules. It can also be used to refine scheduling constraints to determine time durations between the tasks in a plan.
Termination conditions on plan	bp = normal	Boolean condition on patient record. If the condition holds then the plan will terminate and its postconditions will be deemed to hold.
Abort conditions on plan	trend(bp) = falling	Boolean condition on patient record. If the condition holds then the plan will abort and its postconditions will be assumed not to hold.

**Table 2: Attributes and example values of a PROforma plan**

Candidates !	medication({cimetidine, ranitidine})	The set of options among which the decision procedure is designed to choose. These can be defined extensionally (as an explicit set, as here) or intensionally (by rule).
Sources	liver_disease, renal_disease	Data sources which are relevant to making the decision where these are not implicit in the conditions of arguments.
Argument rules or schemas	diagnosis = oesophagitis and liver_disease = absent -> cimetidine : +	Argument schemas have the <i>form</i> of rules in first-order logic but not the categorical <i>force</i> of such rules. An argument rule can support (+) or oppose (-) a decision option, establishing a preference order on the options as a function of the overall balance of pros and cons. The final definition of <i>PROforma</i> will permit the inclusion of quantitative weights on schemas.
Commitment rules or schemas !	preferred(medication = M) and sinister_features = absent	Commitment may be made solely on the basis of relative strength of argument (preference). However in some circumstances, when there are additional safety constraints for example, preferences may not be sufficient. The commitment language permits the designer to define decision rules which define additional logic conditions which must be satisfied before commitment is permitted.
Mandatory data	liver_disease	A standard constraint on all commitments is that mandatory data must have been obtained before the decision can be taken.

**Table 3: Attributes and example values of a *PROforma* decision**

Method !	1. "Give ibuprofen, 10mg" 2. call(print(leaflet1))	The default method for an action is to issue a message to the user interface (1), but an application designer can also include a call to an external procedure where appropriate (2).
Confirmation !	"No" or "Yes" with an authorisation condition.	If a task does not require confirmation it will be assumed to have been done as soon as the request is issued. Otherwise it will wait for confirmation from an authorised source (e.g. a specific user or appropriate grade of staff).
Context, special conditions	"Please explain instruction leaflet to patient and confirm"	The default context of an action is a message to the user interface. If the method is a procedure call, however, the context may define bindings of any variables in the call.

**Table 4: Attributes and example values of a PROforma action**

Enquiry name	Height	Identifier of the parameter which is to be obtained.
Data definition	Type: integer Range: [100:230] Units: [cms] Caption: "height of patient" Default: none Abstraction: between(150-200) -> normal	Set of validation (type, range), metadata(units, caption) and interpretation rules (default, abstractions) for obtaining parameter values.
Method	use_form(height_and_weight)	Default method for an enquiry is to generate a dialogue message. Alternatively a message to an external procedure can be used to obtain the data.
Context, special conditions	"Patient should remove shoes and stand on scales"	The default context of an enquiry is a message added to the default dialogue. If the method is a procedure call, however, the context may define bindings of any variables in the call.

**Table 5: Attribute and example values of PROforma enquiries**