A study of PROforma, a development methodology for clinical procedures

Arjen Vollebregt
Annette ten Teije [1]
Frank van Harmelen
Department of Computer Science and Mathematics
Vrije Universiteit Amsterdam
{annette,frankh}@cs.vu.nl

Johan van der Lei
Mees Mosseveld
Department of Medical Informatics
Erasmus Universiteit Rotterdam
{vanderlei,mosseveld}@mi.fgg.eur.nl

Abstract

Knowledge engineering has shown that besides the general methodologies from software engineering it is useful to develop special purpose methodologies for knowledge based systems (KBS). PROforma is a newly developed methodology for a specific type of knowledge based systems. PROforma is intended for decision support systems and in particular for clinical procedures in the medical domain.

This paper reports on an evaluation study of PROforma, and on the trade-off that is involved between general purpose and special purpose development methods in Knowledge Engineering and Medical AI. Our method for evaluating PROforma is based on re-engineering a realistic system in two methodologies: the new and special purpose KBS methodology PROforma and the widely accepted, and more general KBS methodology CommonKADS.

The four most important results from our study are as follows. Firstly, PROforma has some strong points which are also strong related to requirements of medical reasoning. Secondly, PROforma has some weak points, but none of them are in any way related to the special purpose nature of PROforma. Thirdly, a more general method like CommonKADS works better in the analysis phase than the more special purpose method PROforma. Finally, to support a complementary use of the methodologies, we propose a mapping between their respective languages.

1 Motivation

Over the years, much research in AI & Medicine has been concerned with developing methodologies for applying AI to medical knowledge [11]. This research parallels much work in both Knowledge Engineering and Software Engineering.

In the literature of software engineering many methodologies are proposed for building complex systems, such as OMT [14] or MSA [22]. These methodologies enables us to develop software in a structured way.

Knowledge based systems are of course a particular kind of software systems, and research in Knowledge Engineering in the last decade has shown that it is useful to develop methodologies which are specialised

[1] Supported by the Netherlands Computer Science Research Foundation with financial support from the Netherlands Organisation for Scientific Research (NWO) project: 612-32-006

for this type of systems. Examples of special purpose Knowledge Engineering methodologies are CommonKADS [20, 16], MIKE [1], Generic Tasks [3], DESIRE [17]. In particular CommonKADS is one of the methodologies that is accepted both in research and industry. Many variants based on the CommonKADS methodology are used in industry [9, 2].

PROforma [6] is a methodology for an even more specific type of systems, namely for *decision support systems* and in particular for clinical procedures in the medical domain. This domain is of course more narrow than Knowledge Engineering in general, although clinical procedures in the medical domain still constitutes a large class of different problems (e.g. diagnosis, monitoring, treatment planning). Furthermore the medical domain is a large domain, containing many different types of knowledge (causal, temporal, uncertain, etc.), each of which concerns large volumes of knowledge.

PROforma is intended for supporting the complete process from early knowledge acquisition to the construction of an executable system. PROforma contains a semi-formal language for describing medical decision-making procedures, a tool for constructing knowledge-level models of such procedures, and a tool for executing such model. The fact that the model can be executed is an important characteristic of PROforma methodology. The PROforma methodology is in development at the Imperial Cancer Research Fund in London. Under its new name Arezzo, PROforma is currently proposed as a European standard for specifying clinical procedures.

In this paper, we are concerned with comparing the special purpose PROforma methodology with a more general purpose methodology from Knowledge Engineering, namely CommonKADS. Such a comparison is important for a number of reasons. First, PROforma is a newly proposed method, so it is relevant to compare PROforma with some of the existing alternatives, in particular with a method like CommonKADS which has already found widespread acceptance in practical applications. Secondly, PROforma is a special purpose method, aimed at a narrower (albeit still very broad) application area than CommonKADS. As a consequence, a comparison between PROforma and CommonKADS might tell us something about the trade-off that is involved between general purpose and special purpose development methods in Knowledge Engineering and Medical AI.

The structure of this paper is as follows: In section 2, we briefly outline the approach we have taken to this comparison study. This is followed by two sections devoted to a brief description of both PROforma and CommonKADS (sections 3 and 4). We then describe the actual case-study we performed (section 5) and the conclusions we draw from this case-study (section 10).

2 Evaluation Method

Evaluation of Knowledge Engineering methods is a hotly debated topic in the Knowledge Acquisition community [4, 8, 15, 10]. The main problem of such an evaluation is the difficulty of designing good experiments for such an evaluation, because there are too many variables that play a role. For instance, the background of the users involved in the experiment, ambiguous and/or incomplete requirements on the systems that must be built in an experiment, learning effects with users who repeatedly perform (parts of) an experiment, etc.

For our comparison-study of PROforma and CommonKADS we have chosen for a re-engineering experiment: rather than building new systems with the two methods and comparing the results, we have chosen to compare the efficacy of PROforma and CommonKADS by re-engineering an existing system in both methods. The advantage of such a re-engineering experiment is that the specification of the system which is to be built using PROforma and CommonKADS is perfectly clear, because behaviour of the system is already available. This eliminates at least some of the problems which have hampered other evaluation experiments [8, 15].

As we will describe in section 5, our re-engineering study concerns a realistic system which is already

in fielded use with a large number of general practitioners in The Netherlands. This ensures that we are applying the methods to a realistic case, thereby avoiding the problems of other evaluation experiments which were aiming at artificial (or even toy) problems [8].

Finally, we also tried to minimise noise in the experiment with respect to the user's background. The system is modeled by an intended-user of both methodologies (a person knowledgable in AI and Knowledge Engineering techniques). Furthermore, this person was a novice in both methodologies, ensuring that no hidden learning effects were introduced.

Notwithstanding all this, our experiment is far from ideal. The most obvious limitation is that we have only performed a single re-engineering experiment (a single existing application which was re-engineered by a single person). As has been observed before [13], the size and cost of these evaluation studies in terms of required calender-time and man-power prohibits larger scale evaluations. Secondly, our evaluation study was performed using a rather early version of the PROforma tools. We will have more to say about the effects on this in our conclusions (section 10).

3 The PROforma methodology

Several attempts have been made to design knowledge representation systems that make it possible to support clinical procedures. As also mentioned in [6] the current techniques have limitations. The current techniques are not reliable or expressive enough. For example there is often no support to deal with uncertainty or temporal aspects. The PROforma method is designed to meet these problems. The most important element of the method is the PROforma language, this is a knowledge representation language which has, among others, decision and process management tasks. The PROforma language is also a specification language: a declarative formalism that can be used to design software.

How to use PROforma Building a clinical protocol in PROforma is done in two phases. In the first phase a high level description of the protocol is made with the graphical editor. In the second phase the graphical design is instantiated with knowledge needed for the enactment of the protocol.

The 'task' is the top level structure in PROforma, all the other structures are derived from it. The PROforma language is designed to give support to a wide variety of clinical tasks, like decisions, scheduling of actions over time, monitoring etc. PROforma supports four basic classes of tasks (see figure 1):

- plan: a sequence of several sub-tasks or components. Plan components usually have an ordering to specify temporal, logical or source constraints.
- decision: is used where a candidate must be chosen from a given set using arguments pro and contra.
- action: a procedure that has to be executed outside of the computer system, like an injection.
- enquiry: requests information needed to execute a certain procedure.

Such protocols can be constructed with a graphical editor, as shown in figure 1

The protocols made with the graphical editor have to be instantiated with specific data, such as pre-conditions, post-conditions, abort-conditions and data definitions. The different tasks in PROforma have a common internal structure (common task attributes) and a specific structure (task specific attributes). For the common task attributes so called templates are available, which ease the entering of data. These templates are equal for all tasks. Each task has the following attributes (these tables are based on [5]):

Common attributes of tasks

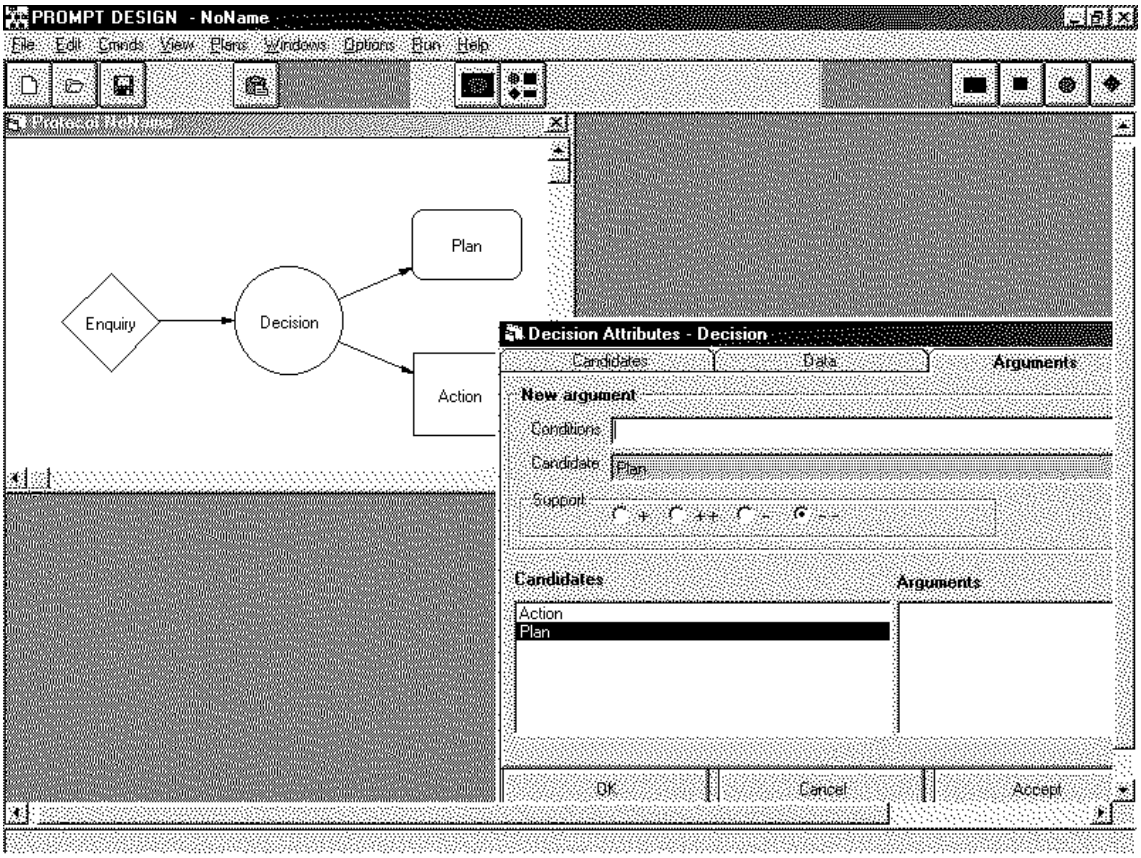

Figure 1: A graphical representation of PROforma plans in the PROforma editor: diamonds are enquiries, rounded rectangles are plans, normal rectangles are actions, and circles are decisions. The arrows indicate scheduling constraints.

Attribute	Description
Title	Descriptive title of the task
Identifier	Identifier name and parameters
Description	Textual description of the task
Goal	Goal of the task
Trigger conditions	Conditions that have to be met before a task can be started outside the scheduling constraints. The trigger conditions can start a task without meeting the scheduling constraints of its task.
Preconditions	Conditions that have to be met before a task can be started.
Postconditions	Conditions which become true when a task is finished
Version info	Author, date of design, modifications etc.

As mentioned above, every task has its own specific attributes:

Specific attributes of a plan task

Attribute	Description
Components	List of the subtasks
Scheduling constraints on subtasks	Constraints which specify the ordering of the tasks
Temporal constraints on subtasks	Constraints which specify the timing of the tasks
Optional tasks	List of tasks that are optional
Non confirmatory tasks	Tasks that do not require confirmation
Number of cycles	Number of times a tasks should be repeated
Cycle until conditions	On which condition should the repetition stop
Repetition interval	Time interval between the repetition of the task
Abort conditions	Conditions which abort a plan when they are met
Terminate conditions	Conditions which specify when a plan is finished

Specific attributes of a decision task

Attribute	Description
Candidates	Candidates for the result of the decision
Arguments	Rules with arguments for the various candidates
Commitment rules	Rules to choose a certain candidate
Sources	Information sources which are needed
Mandatory Info	Information needed to confirm a decision

Arguments: The pro's and con's of a candidate are specified using the arguments. There are four different possibilities:

$+$: the support for a candidate is increased with one argument.
$-$: the support for a candidate is decreased with one argument.
$++$: the candidate is certainly chosen.
$--$: the candidate is certainly not chosen.

The decision is made in the following simple way: the support value of a candidate is its number of $+$'s minus its number of $-$'s. The candidates are ordered based on their support value, but a $++$ or $--$ always has the highest priority. Using the commitment rules a decision is made. This decision is made using a so called threshold: when for a certain candidate a certain threshold is reached the candidate gets chosen. When different candidates have the same amount of arguments the first one is chosen.

Specific attributes of an action task

Attribute	Description
Procedure	The procedure that has to be executed
Context	Context, special conditions etc.

In the current version of PROforma the action procedure is only displayed on screen. The user has to execute the procedure. In the future some actions could be executed automatically, by calling external software.

Specific attributes of an enquiry task

Attribute	Description
Data required	Data item(s) required. The data items are for example requested from the user or host system.

PROforma has a data expression sub-language that allows the constructions of arbitrarily complex expressions and conditions. This language can be used wherever expressions or conditions must be specified, e.g.

preconditions, decision argument rules. Data items may have multiple values, which can be manipulated as a set of values, and can also be manipulated by specifying temporal aspects of the data. The data language incorporates all the usual arithmetic operators, as well as various set functions like sum, average, maximum and minimum. Furthermore temporal predicates and various comparison operators are provided.

Data definitions enable PROforma to communicate with external databases or host systems. With the help of data definitions it is possible to define for the data items from the protocol what the data type of the data item is, which values it can accept and where the data comes from. The data definitions are specified in a separate file that is loaded when starting the protocol.

The version of PROforma used in our case study was an evaluation copy available to interested parties. Currently, a new release of PROforma is prepared under the commercial name "Arezzo".

PROforma has been used to develop a number of clinical guidelines, including several for use in primary care. Several GPs in a London surgery have been doing a study using a guideline for Dyspepsia written in PROforma. A guideline for managing pain control has been developed in collaboration with St. Christopher's Hospice in London. Another research project at the ICRF, called RAGS, has used PROforma for decision support in assessing cancer risks. The European project MACRO has embedded PROforma in its clinical trials application, which is currently in use in at least four different centres in Europe.

It is hoped that Arezzo will become the standard for representing guidelines in medicine. We have plans for commercial exploitation and are already in consultation with potential commercial partners. the ICRF expect to formally publish the Arezzo Language soon, and meanwhile we continue further development.

4 The CommonKADS methodology

The CommonKADS methodology is a structured approach to analysis, design and management of knowledge based systems [20], [16]. CommonKADS is based on the following principles: a *model driven approach* (since models are purposeful abstractions); *knowledge-level principle* (modelling knowledge independent of implementation details); *limited-role concept* (structuring problem solving by identifying knowledge -types).

CommonKADS consists of a number of models:

- Organisation Model: concerned with the organisation in which the KBS must function
- Task Model: modelling inputs, outputs, preconditions and performance criteria of the global task
- Agent Model: assigning tasks to various agents such as humans, computers, KBS, etc.
- Knowledge-Model: a detailed but implementation independent description of the knowledge used to perform the desired tasks, and the role that different knowledge components play in problem-solving.
- Communication Model: stating the communication between the various agents.
- Design Model: describing the technical specification of the system.

A CommonKADS knowledge model consists of three categories of knowledge (often referred to as "layers"): domain, inference and task knowledge (for the purposes of this paper we ignore the strategic knowledge). The first category contains a description of the *domain knowledge* of a KBS application. It describes objects properties, relations, concepts etc in the application domain. This description should be as much as possible independent from the role this knowledge plays in the reasoning process. The *inference knowledge* of a CommonKADS knowledge model describes the reasoning steps (or: inference actions) that can be performed using the domain knowledge, as well as the way the domain knowledge is used in these inference steps. The inputs and outputs of such inference steps are called "knowledge roles". CommonKADS represents the relations between the inference steps through their shared input/output roles, with a dependency graph among inference steps and knowledge roles. Such a graph, called an inference structure, specifies

only data dependencies among the inferences, not the order in which they should be executed. The use of domain-knowledge by inferences is described by mapping such domain-knowledge to input-roles of inferences. The execution order of the inference steps is specified as *task knowledge*. For this purpose, CommonKADS uses a simple procedural language with procedural decomposition to execute inference steps and predicates to test the contents of knowledge roles. These procedures can be organised in a task-decomposition tree, and can be combined using sequences, conditionals and iterations. The CommonKADS architecture is illustrated in figure 2.

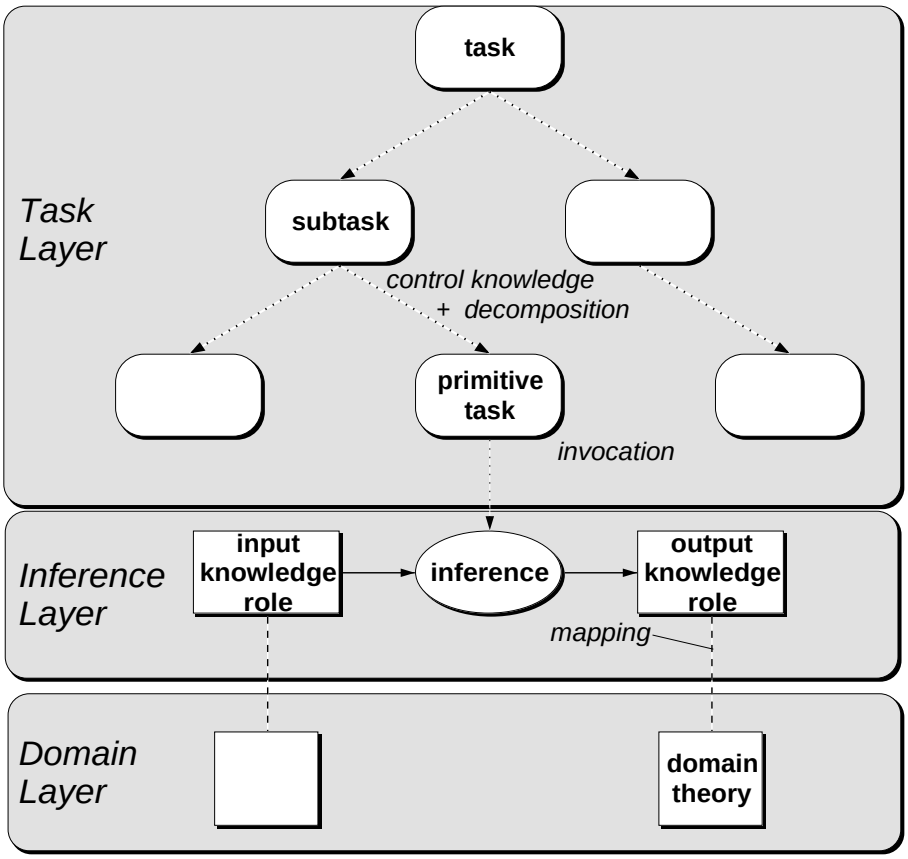

Figure 2: structure of the CommonKADS knowledge model

As shown in figure 2, the CommonKADS architecture consists of separate layers, with explicit connections between these layers. Primitive procedures at the task-layer are mapped to inference steps at the inference layer, and knowledge-roles at the inference layer are mapped to knowledge at the domain layer. These separations were introduced to maximise the possibilities for re-use. For example, a combination of task- and inference layer that model a reasoning strategy for treatment planning could be reused and applied to different sets of domain knowledge (e.g. treatment planning for heart diseases or asthma). Vice versa, the same domain knowledge (e.g knowledge about asthma) could be used for different inference- and task-layers (e.g. to either diagnose or to treat heart diseases).

5 The case-study BloedLink

5.1 BloedLink

In The Netherlands general practitioners (GPs) yearly spend 75 million guilders on lab research [12], and the GP often does not have enough knowledge about the right indications to choose the various tests. Recent data from the National study of diseases and actions in the office of the GP's shows a large variation per working hypothesis in the requests of tests [7]. On average, as much as 45 different combinations of tests are requested per working hypothesis Excessive request of lab research does not only generate unnecessary costs, but can also be the cause of deviating results (e.g. false positives), which can by themselves be the reason for more (unnecessary) diagnostic activity. One way of altering the request behaviour of GP's is to alter the request form. There are two strategies for doing this:

1. The use of a restricted request form, on which rarely used tests have been left out.
2. Problem oriented requests. A form on which for the specific problem, only the related tests are shown. (these were electronic forms).

The Erasmus University carried out research with 60 GP's in the region Delft/Westland/Oostland using BloedLink with the GP Information System 'Elias'to find out how many tests were included in each lab request [19]. The use of the restricted request form showed an average of 6.9 tests per request, whereas the problem oriented request form showed an average of 5.5 tests per requests. On average it turns out that there is a decrease 20% in tests per request when the problem oriented form is used. These figures can be explained by the fact that the problem oriented form is a further restriction on the restricted form, since the problem oriented form is based on a specific problem, excluding not only rarely used tests, but also requests that were irrelevant for the specific problem.

The problem oriented request forms have to be based on a standard. Research has shown that from the guidelines of the Dutch College of General Practitioners (DCGP) concerning blood research, unambiguous advice associated to working hypotheses can be generated [18]. In other words, problem oriented requests can be derived from the DCGP-guidelines. A possibility is to use an electronic form, based on the DCGP-guidelines, to present working hypotheses and the associated requests. Since GP's are increasingly using the electronical patient record (EPR) in daily practice and the EPR is very useful to support protocol based care, this is an interesting option. By incorporating the decision support programs in the EPR, information like DCGP-lab guidelines can be made directly available to the GP. These results have been the main motivation for developing a module that supports the GP with requesting blood tests directly from the EPR. This module is called BloedLink. BloedLink shows the right laboratory protocol indexed to the reason of request, based on a working hypothesis chosen by the GP. An (optional) help text (as much as possible an abstract from the text of the related DCGP- standard) gives information on the selection in the protocol. The module gives the GP the opportunity to deviate from the protocol by leaving tests out or adding tests to the protocol. When the GP has finished the request form it is either printed or sent to the laboratory electronically (although at the moment the latter option is supported by only a few laboratories).

BloedLink is a decision support system that is integrated with the electronic patient record of two systems (Elias and MicroHIS). BloedLink operates as follows: When the GP wants to request blood tests, she activates the system. The system queries the GP about the reasons for requesting the tests. The system continues asking questions until the working hypothesis is identified. To determine the working hypothesis, the GP has to answer up to four questions that deal with the reason for requesting blood tests. When the working hypothesis is identified, the system proposes the relevant tests.

The current version of BloedLink has some limitations:

- *superfluous information is presented to the general practitioner*
  For example if a hypothesis is not confirmed, one shouldn't have the possibility to choose a monitoring protocol, but only confirm or exclude the hypothesis. Another example is that when a patient is male the guideline about pregnancy shouldn't be shown.
- *no support for monitoring*
  In the current version of BloedLink when the working hypothesis turns out to be a monitoring task, the GP is responsible for remembering when certain tests should be administered. BloedLink should also prompt the GP the for appropiate tests at the right moment.

Besides re-engineering the existing BloedLink system we have decided to deal with these limitations in the following ways:

- *using knowledge about the patient*
  To prevent the presentation of superfluous data to the GP, BloedLink should use data about the patient to determine which data should be presented. This can be done by querying the electronic patient record for data about the patient. The new version of BloedLink should make use of more advanced reasoning methods.
- *support for monitoring*
  BloedLink should support monitoring. The current version of BloedLink does not support automated monitoring. The new version of BloedLink should have this possibility.

The proposed extensions to BloedLink are included in the new version of BloedLink.

6 CommonKADS model of BloedLink

This section discusses the three layers of a CommonKADS knowledge-model of BloedLink. We discuss the model top-down, starting with the task-layer, followed by inference and domain layer.

6.1 Task layer

In figure 3 the task structure of BloedLink is presented. This task structure is a task decomposition tree. The rounded boxes are tasks, tasks are defined in terms of their input and output. The squared boxes are task-methods. A task-method describes how a task can be realized through a decomposition into sub-functions plus a control regime over the execution of the sub-functions [21]. The task and the task-method can be best understood as respectively the "what (needs to be done)" and the "how (is it done)" view on reasoning tasks [21]. The combined tasks (fig. 3) carry out the task of BloedLink, finding the right set of tests for an abstract hypothesis.

An example of a task description is:

```
task find tests;
     domain name : find-test-by-select-and-match;
     goal : "find the right tests with hypothesis and objectives";
     roles:
        input:  abstract-hypothesis: "abstract hypothesis from user";
                objectives: "the objective for performing the tests";
        output: tests: "the adviced set of tests for the hypothesis";
end task find tests;
```

Below we describe the goals of each of the tasks that correspond to the squares in the decomposition tree (fig. 3) and to the rounded boxes in the inference structure (see fig. 4).

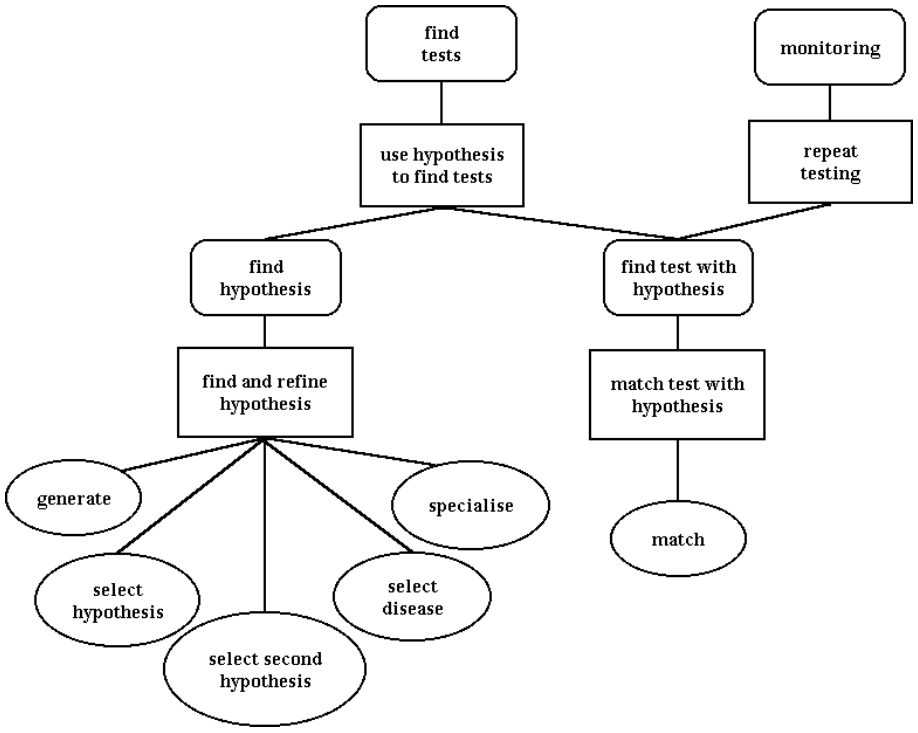

Figure 3: The task layer of BloedLink

TASK	GOAL
find tests	find the right tests using an abstract hypothesis and objectives
monitoring	monitor the treatment by applying the right tests over time. For example, to check the glucose level after four weeks of treatment.
find hypothesis	find hypothesis with abstract hypothesis, patient history and selection of user. For example: ”confirm monocytair anemia”
find test with hypothesis	find adviced set of tests for concrete hypothesis

We give two examples of the control structure of the taks-layer of BloedLink: the task control (task-method) of “monitoring” and of the task control of “find tests”. The first one specifies an iterative structure over a subtask, the second a sequential structure.

```
task-method repeat testing;
   realizes:       monitoring
   decomposition :
      tasks : find test;
   roles:
   intermediate:
   control structure:
      while time < set-time do
         find test;
      end while
end task-method repeat testing;

task-method use hypothesis to find test;
   realizes:       find test;
   decomposition :
```

```
      tasks : find hypothesis, find test with hypothesis;
   roles:
   intermediate: find hypothesis
   control structure:
     find hypothesis(abstract hypothesis + select hypothesis + patient
     history + patient data + select objective -> concrete hypothesis);
     find test with hypothesis(concrete hypothesis -> tests);
end task-method use hypothesis to find test;
```

6.2 The inference layer

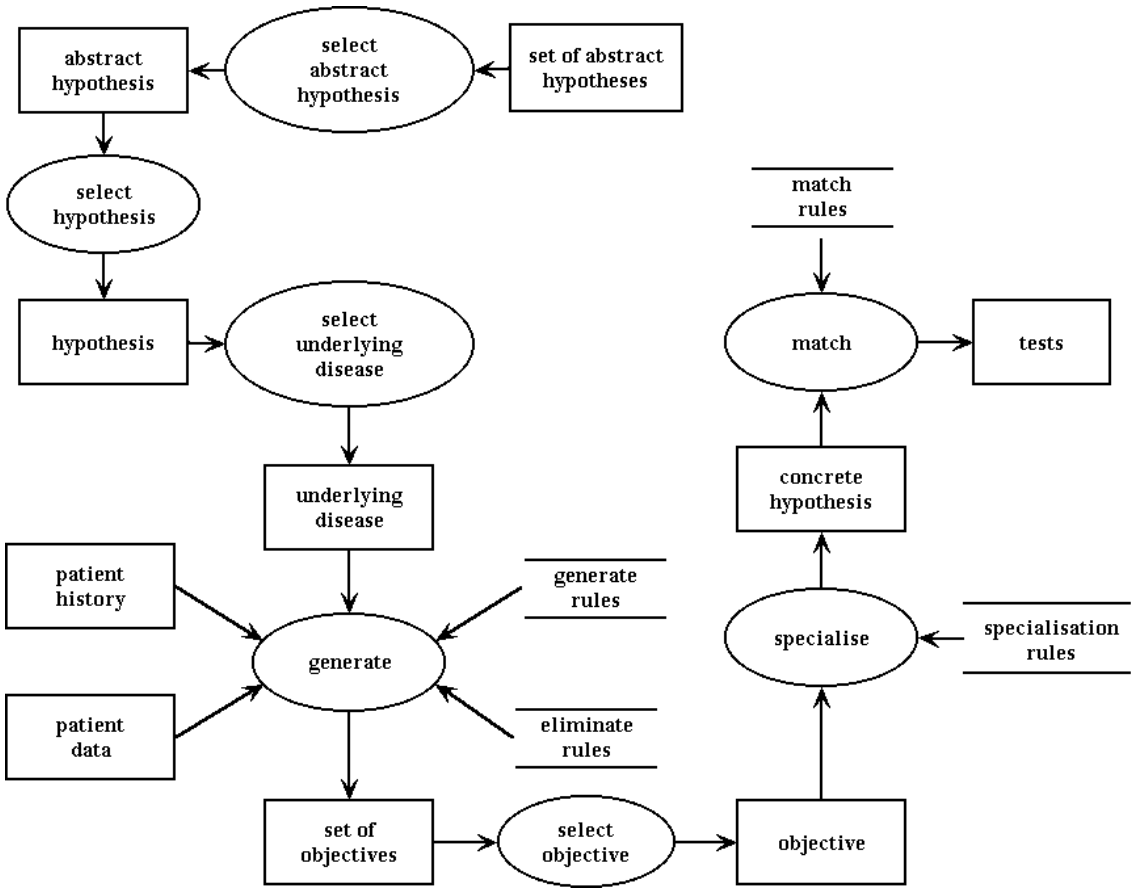

Figure 4: The inference structure of BloedLink.

Figure 4 shows the inference structure of BloedLink. The boxes are the input and output roles, the ellipses are the inferences. The items between to horizontal lines are the rule schema's of the domain layer which are used in the inferences. Only the first inference "select abstract hypothesis" is described in detail below as illustration.

```
inference select-abstract-hypothesis;
   roles: input abstract-hypothesis;
          output abstract-hypothesis;
          dynamic obtain;
```

```
   domain-mappings:
          abstract-hypothesis -> abstract-hypothesis;
   specification
          "User selects an abstract hypothesis based on a set of
           abstract hypothesis";
end inference select-abstract-hypothesis;
```

The user selects an abstract hypothesis based on a set of abstract hypotheses. The domain mappings specify the concepts of the domain that are mapped onto roles.

A brief description of the other inferences is as follows:

- *inference select-hypothesis*
  The user selects a hypothesis based on an abstract hypothesis.
- *inference select-underlying-disease*
  The user selects an underlying disease from a set of underlying diseases generated from the hypothesis.
- *inference generate*
  Objectives are generated based on the hypothesis, patient history and patient data.
- *inference select-objective*
  The user selects an objective from a generated list of objectives.
- *inference specialise*
  A concrete hypothesis is selected based on objective, patient data and the hypothesis.
- *inference match*
  The inference 'match' selects tests using concrete hypothesis and objective.

6.3 The domain layer

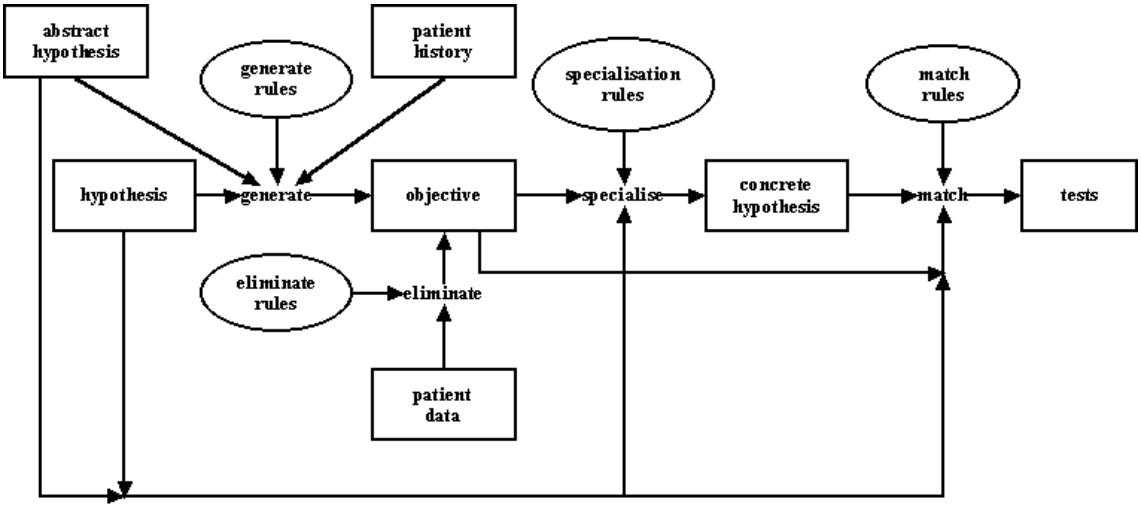

Figure 5: The domain structure of BloedLink.

Figure 5 shows the domain structure of BloedLink. Here the data dependencies of the inferences are shown, e.g. which domain knowledge is used for which inference. The boxes are the concepts, the ellipses the rule -schema's and the items without a border around it are the inferences. The concepts are the data for BloedLink and the rule-schema's the operations on the data. The inferences correspond to the inferences

of the inference layer. For example the inference ”generate” uses the concepts “abstract hypothesis”, “hypothesis”, “patient history” and the rule-schema “generate rules”. An example of a concept can be found below.

```
concept abstract hypothesis;
   "abstract hypothesis selected by user"
properties
   abstract hypothesis: liver, thyriod hypercholesterolemy, etc.;
end concept abstract hypothesis;
```

Below is an example of a rule schema.

```
rule-schema generate-rules;
antecedent:
    hypothesis;
    cardinality : 1+;
consequent:
    objective;
    cardinality : 1;
symbol:
    generate;
examples:
    abstract_hypothesis = liver_diseases and hypothesis = hepatitis_B
    indicates objective = confirm;
end rule-schema generate rules;
```

In this section we have given an impression of the CommonKADS model of BloedLink of our case study.

7 PROforma model of BloedLink

In this section we present the PROforma specification of BloedLink.

Figure 6 below gives a high level overview of the BloedLink protocol.

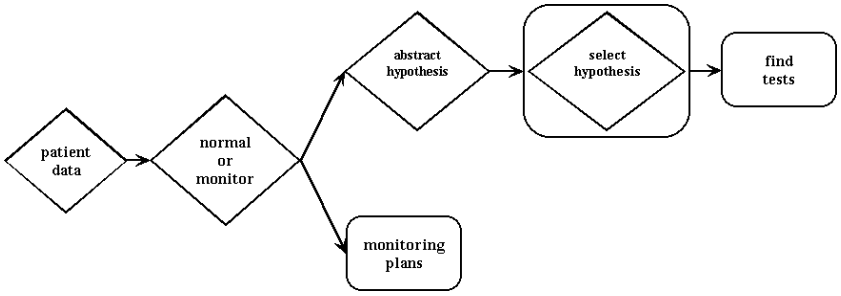

Figure 6: The “BloedLink” Plan: High level PROforma structure for the BloedLink protocol.

First of all the patient data is requested: age and sex of the patient. These will be stored in the task attributes “age” and “sex”. The patient data will be requested from the host system. Then the user is asked if he wants to monitor the disease of a patient directly or wants to confirm/exclude a disease first. In the latter case, what follows is an enquiry “abstract hypothesis”, a plan containing only the enquiry “select hypothesis” and the plan “find tests”. The enquiry “abstract hypothesis” requests for an abstract hypothesis. The enquiry “select hypothesis” is put inside a plan to make sure that the protocol continues even if the precondition of

the enquiry ("are there hypotheses for the selected abstract hypothesis?") is not true, since a plan always succeeds. The enquiry requests for a hypothesis. Only the hypotheses that are appropriate are presented. Finally a plan "find tests" is started to find the adviced tests for the hypothesis.

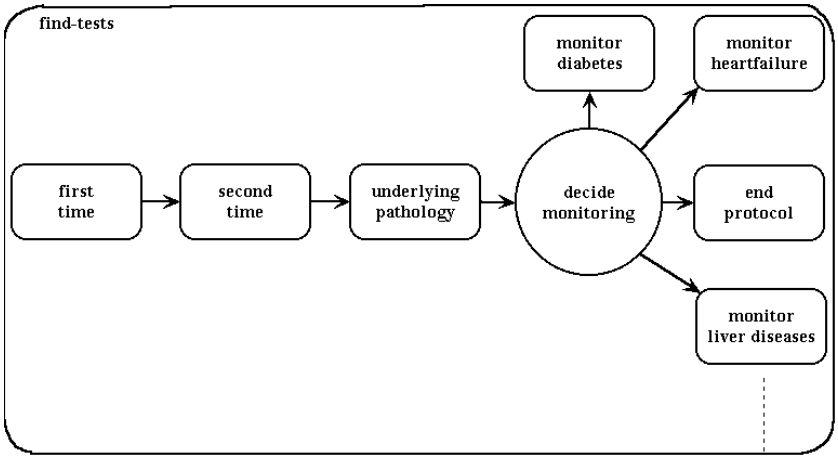

Figure 7: The components and scheduling constraints of "find tests" plan. The "find tests" is a subplan of "BloedLink" plan.

Figure 7 shows the "find tests" plan. The subplans of "find tests" are discussed separately below. First a plan "first time" (figure 8) is started to find the tests that are adviced when no tests have been performed previously. The decision "decide monitoring" decides which monitoring plan should be started.

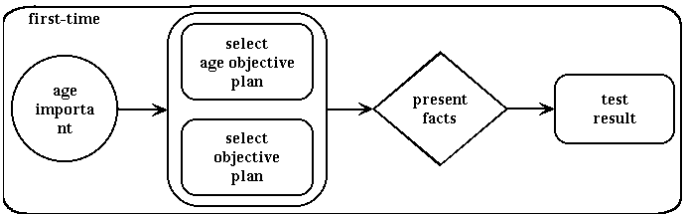

Figure 8: The "first time" plan. The "first time" is a subplan of "find tests", which is itself a subplan of "BloedLink" plan.

The decision "age important" decides whether the age of the patient is important for the next selection of an objective. If the age is important the protocol will select the plan "select age objective plan" that presents the objectives that are appropriate for a patient with the specified age. The selected objective is stored in the data item "objectivea". If the age is not important the protocol will select the plan "select objective plan". that does the same, but stores the selected objective in the data item "objective". Later on the data item "objectiva" is copied to the data item 'objective' to allow the remains of the protocol to refer to the data item "objective". The enquiry "present tests" presents the tests that are appropriate given the selected objective. Finally the plan "test result" is started. It requests for the test results of the selected tests.

After the first time plan the second time plan (figure 9) is started. For some guidelines like "Dysfunction of the thyroid gland" and "Asthma-COPD" a second step is needed because if the test results of the first step were abnormal a second test is needed to confirm the hypothesis. The protocol advises a different hypothesis if the results of the second time plan conflict with the chosen hypothesis. If for the chosen hypothesis an underlying disease could be the cause of the illness, the underlying pathology plan is started:

In the underlying pathology plan (figure 10) the general practitioner is asked if she/he suspects an underlying disease. If so the user is asked to select an underlying disease. Only the diseases that are appropriate at

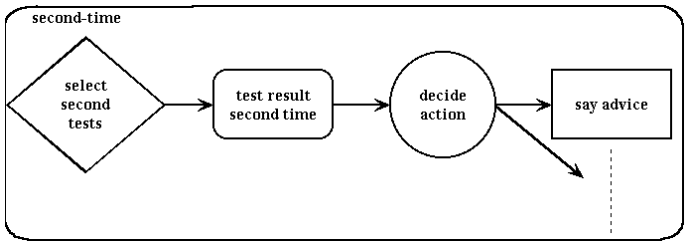

Figure 9: The “second time” plan. The “second time” plan is a subplan of “first tests”, which is itself a subplan of “BloedLink”.

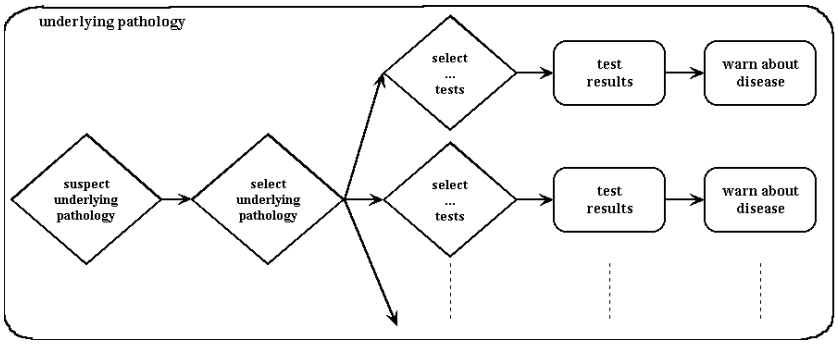

Figure 10: The “underlying pathology” plan. The “underlying pathology” is a subplan of “find tests”, which is itself a subplan of “BloedLink” plan.

the moment are presented. Then the tests needed to confirm the underlying disease are presented. The user can deselect tests if she/he wants to. Next the test results are requested and if the test results confirm the underlying disease the user is warned that the selected underlying disease is the probable cause of the illness. The user can proceed with the current protocol though. The user can also select another underlying disease to test. After the underlying disease plan the decision “Decide Monitoring” decides if a monitoring plan is appropriate for the current hypothesis. If so a monitoring plan is started (for instance monitor ”heart failure” of fig. 7), if not the protocol ends.

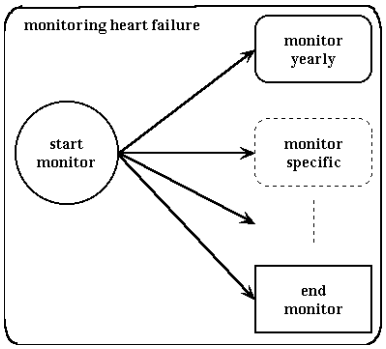

Figure 11: An example of a monitoring plan (for instance “monitor heart failure”). Such a monitoring plan is a subplan of “find tests”, which is itself a subplan of “BloedLink” plan.

If in figure 6 the user chooses for the monitoring branch, the decision “start monitor” of figure 11 decides which monitor tasks to start or, if no monitoring is needed, to end the protocol.

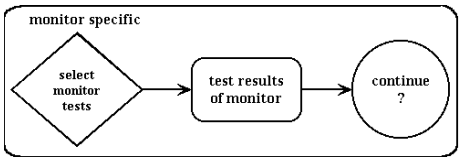

Figure 12: A specific monitoring plan (for instance “monitor yearly” or “monitor specific”) of a particular protocol (for instance “monitor heart failure”). Such specific monitoring plan is a sub-sub plan of “find tests”, and thus a sub-sub-sub plan of “BloedLink”

Figure 12 shows a monitoring plan for the plan chosen in figure 11. The enquiry “select monitor tests” presents the tests that are adviced for the monitoring task and stores them in a specific data item for that disease. The plan “test result of monitor ...’ is generally the same as the plan “test result” of the “first time” plan. The only differences are the names of the data items and tasks. The decision “continue” decides whether the monitoring task should continue or end because the goal of the protocol is reached.

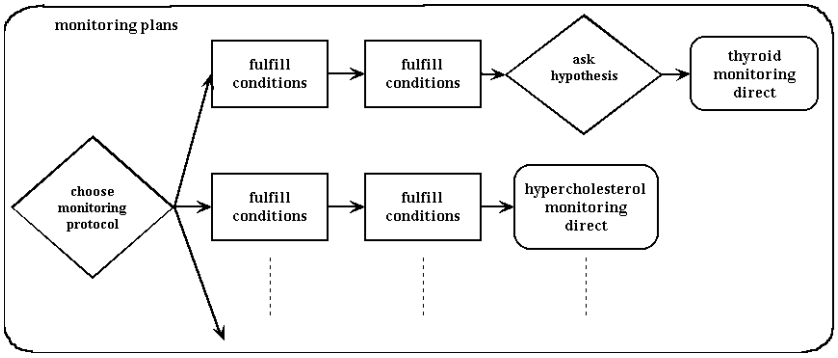

Figure 13: The “monitoring plans” is a subplan of BloedLink plan.

Figure 13 shows the plan for “monitoring plans” of figure 6. First the user is asked which monitoring protocol he wants to be started. This is done with the enquiry “choose monitoring protocol”. Then certain conditions have to be fulfilled. The data item “abstract hypothesis” with the correct value has to be asserted in the PROforma database because the monitoring plans use this data item.

8 Evaluation of PROforma

In this section we will discuss the conclusions concerning both methods that can be drawn from the experiment described above. We will first discuss a number of shortcomings that we observed in PROforma and CommonKADS (section 8.1). It will turn out that the methods have rather complementary properties, leading us to the suggestion that they should be combined in different stages of the Knowledge Engineering process (section 9). Such a combination requires a mapping between the two methods, for which we will outline a proposal.

8.1 Shortcomings of PROforma

We have identified five mayor shortcomings of PROforma as follows:

1. necessity of using global data representations

2. data-dependencies are difficult to track
3. no overview of the global structure
4. some missing functionality
5. performance problems

Necessity of using global data representations PROforma contains no mechanism for encapsulating data locally within a task. Every variable (data item) is global. This means that all data of one task can be overwritten by another task, for example data used in pre- and post-conditions of tasks. This is contrary to commonly accepted principles in Software Engineering, which promote local encapsulation of data as much as possible, in order to prevent problems such as name-clashes, illegal read- and write-access. This is all the more true since PROforma contains built-in predicates to add and remove such global data without any protection. This makes it very hard to develop independent tasks and hampers reusability of PROforma's knowledge components. The only way currently available in PROforma to avoid this problem is to ensure that all data items have globally different names.

Data-dependencies are difficult to track The only dependencies between tasks which are currently visualised in PROforma are the temporal scheduling constraints. However, besides these temporal dependencies, tasks also have important data-dependencies: one task uses data produced by others. For example, "select-hypothesis" from figure 6 produces a hypothesis which is used by "select-objective-plan" occurring in "first-time plan" (figure 8), which is a subplan of "find-tests plan" (figure 7). These data-dependencies are not visualised in PROforma. The only way to view the data used by the task is to inspect all task-specific attributes and all common task attributes associated with a given task. This problem is compounded by the fact that all data-structures are global, as discussed above. Besides indicating *that* a data-dependency exists, it should also be indicated *which* data is involved in this dependency (ie which type of data from one task is used by another). It is easy to imagine how such data-links can be visualised, and this is being considered for future versions of PROforma.

No overview of the global structure PROforma only visualises a protocol at a single level of detail, namely all tasks that occur in the top-level of a plan. When descending into a subplan, the surrounding parts of the protocol are not seen. This makes it difficult to get a global overview of the nested structure of the plan.

Some missing functionality In some places we found some functionality or expressiveness missing from the PROforma language. An example of this is in figure 6. There, the enquiry *select-hypothesis* is put inside a plan only to make sure that the protocol continues even if the precondition of the enquiry ("are there hypotheses for the selected abstract hypothesis?") is not true, since a plan is always guaranteed to succeed. It would have been more natural to specify this fail-safe behaviour directly in the enquiry-task. This, and similar problems are related to the fact that we were using an early version of the PROforma toolkit, and are expected to improve in future versions.

Performance problems Since PROforma is explicitly intended not only as an analysis tool, but also as a delivery vehicle for the final operational system, its computational performance is crucial. In particular, the BloedLink system is intended to be used by general practitioners who have generally only very limited hardware available. Consequently, the current performance of the PROforma system is a problem for its use as delivery platform.

Comparison with CommonKADS It was rather a surprise to us that none of the above points are in any way related to the special purpose nature of PROforma. Secondly, in comparison with CommonKADS, we notice that the above points (1), (2) and (3) are handled better in CommonKADS: CommonKADS provides encapsulation of data within tasks and inferences, data-dependencies are visualised at the CommonKADS inference-layer (the CommonKADS task layer suffers from the same lack of visualisation of data-dependencies), and the global structure of a CommonKADS model is typically depicted as in figure 2.

8.2 Strong points of PROforma

We now discuss some strong points of PROforma, which can conversely be seen as corresponding shortcomings of CommonKADS.

Temporal constraints In medical reasoning, time is often an important aspect (e.g during treatment monitoring). These temporal aspects occur frequently in BloedLink, but CommonKADS has no easy way to represent these aspects. Unlike in PROforma, reasoning in CommonKADS about time-points, time-intervals and time-constraints must be modelled from scratch if required.

Ordering constraints among tasks BloedLink sometimes has to go back to an earlier inference when the chosen option turned out to be incorrect when more information become available. Under certain circumstances, it is also necessary to immediately abort the reasoning (for instance when during monitoring the test results become normal). However, both the graphical notation and the control-language of the CommonKADS task structure assume a continuous control-flow during task-execution, which makes it hard (if not impossible) to represent such discontinous jumps in the control-structure.

Executable model After the graphical PROforma model has been extended with all the required task-attributes, the result is an executable protocol. No further implementation effort is needed to obtain a running system. This is in contrast with KADS, where the knowledge model is not executable, and a significant effort is often needed to transform this to an executable form.

We notice that the first two points (temporal reasoning and non-local control flows) are related to requirements of medical reasoning. It is perhaps here that the advantage of using a more special purpose language like PROforma over a general purpose method like CommonKADS pays off. On the other hand, these features are required for many different domains, and should perhaps simply be regarded as shortcomings of CommonKADS per se.

9 CommonKADS/PROforma combination

In the beginning of our study we considered CommonKADS as a general methodology for knowledge based systems, whereas PROforma is a methodology for a specific type of knowledge based systems. However, it turns out that the methodologies are complementary. Our study has shown that CommonKADS is a methodology that is very useful in the analysis-phase. In this phase the more specific PROforma language is too implementation oriented. The analysis of several types of knowledge, and where these types are used in the various reasoning steps is very difficult in PROforma, because reasoning steps have to be represented as a particular kind of task (e.g. decision, plan) and the instances of the knowledge types must be encoded in the attributes of these tasks. However, in the analysis phase it is not all clear what type of task the reasoning steps should be, and therefore it is even more difficult to encode instances of the knowledge types in the

attributes of PROforma tasks. A more general method like CommonKADS works better in the analysis phase then a special purpose language. This is a result of our study that we did not at all expect.

On the other hand, when we have built a PROforma model then we also have an actual implementation of a system, whereas in CommonKADS we still have to implement the model in some other language. Our study has taught us that CommonKADS and PROforma are complementary: use CommonKADS as analysis methodology and use PROforma after the analysis phase for building a clinical procedure application. It is therefore useful to come up with guidelines for mapping a CommonKADS model onto a PROforma model.

A mapping of CommonKADS into PROforma consists of a mapping of every layer of a CommonKADS model: the task layer, the inference layer and the domain layer. We will discuss each mapping below.

Mapping of the task layer The task layer consists of the task decomposition tree and the control structure. The primitive inferences of the task structure (the leaves of the decomposition tree) have to be translated to plans in PROforma. The control structure specifies the ordering of the tasks. This ordering will in general be the same in CommonKADS as in PROforma model, therefore create scheduling-links between the PROforma plans according to the control structure of the task layer. The result of the mapping of the task-layer is the top-structure of the PROforma model including the scheduling links. The structure of the CommonKADS task-decomposition tree can be modelled by a corresponding nesting of PROforma plans, although this nesting cannot be visualised in the current PROforma graphical environment. Figure 14 and the following control structure illustrate the mapping of a task layer into a partial PROforma model.

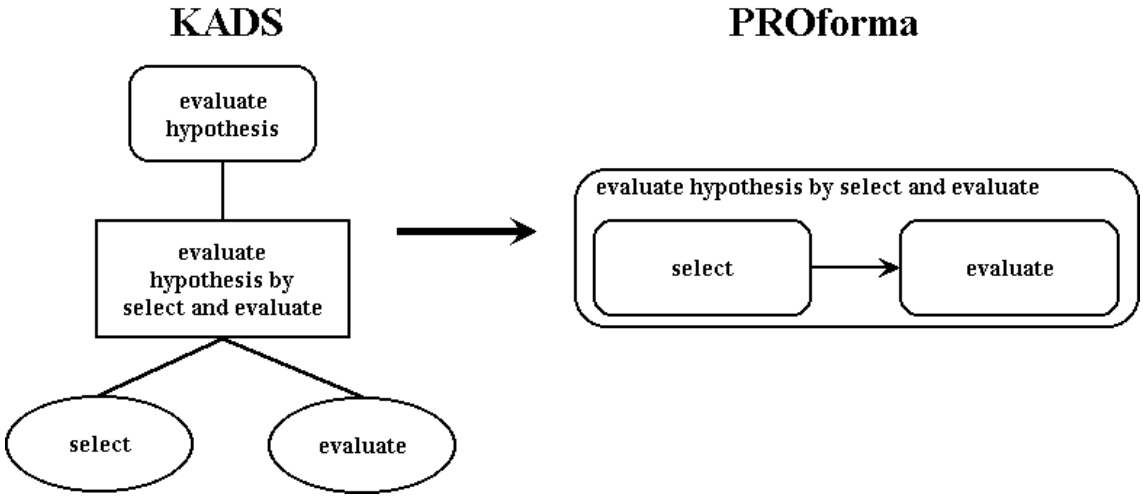

Figure 14: Mapping CommonKADS task layer to PROforma.

```
task-method evaluate by select and evaluation;
....
   control structure:
      select(hypotheses -> hypothesis);
      evaluate(hypothesis -> decision);
end task-method evaluate by select and evaluation;
```

Mapping of the inference layer The PROforma plan made with the task layer is refined with the inference layer. Static roles and input roles of the inferences are translated to enquiry task of PROforma. The instances of the roles become data-items of PROforma task (see figure 15. A CommonKADS inference structure explicitly represents the roles of the domain knowledge and the dependency of these roles

in the several reasoning steps. However, both the abstraction of domain-knowledge to roles and the data dependency between inferences are lost in a PROforma model.

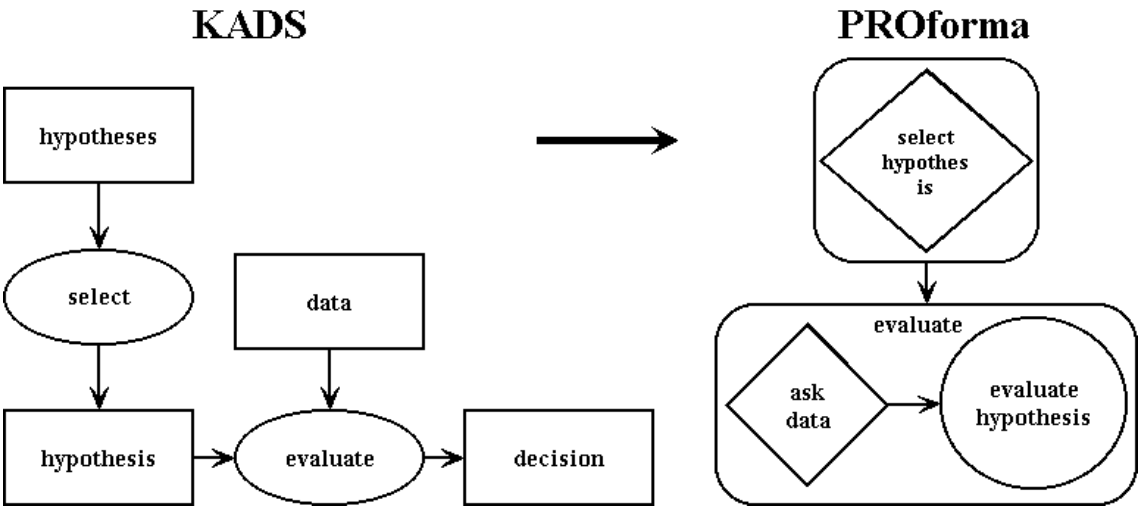

Figure 15: Mapping CommonKADS inference layer to PROforma.

Mapping of the domain layer The concepts of the domain layer map directly to the data attributes of PROforma tasks. A mapping of the domain structure is not possible. The domain knowledge is distributed over several places like templates, data definitions, actions, etc. For instance, rule schemas can map to the templates of decisions, a knowledge base, actions and enquiries. One rule in CommonKADS can be modelled with several tasks in PROforma which have their data stored in different places.

10 Conclusion

In the last decade knowledge engineering has shown that special purpose methodologies for knowledge based systems are useful, besides the more general methodologies of software engineering. This study shows that an even more specialised methodology like PROforma is useful. PROforma is intended for clinical procedures in the medical domain, which still constitutes a large class of different problems (e .g. diagnosis, monitoring, treatment planning) within a large domain.

For our evaluation study, we chose a method based on re-engineering a realistic system in two methodologies: the new and special purpose KBS methodology PROforma and the widely accepted, and more general KBS methodology CommonKADS.

Both the CommonKADS and PROforma methodologies turned out to be useful and easy to learn modelling tools. Both methodologies have their limitations. Problems during the modeling process using the CommonKADS methodology occur only at the task layer, for instance the lack of temporal reasoning. The opposite seems to be the case with PROforma. In the PROforma modeling process problems occurs at the level of the inference- and domain level. As pointed out earlier PROforma could be improved at several places in particular: the lack of data encapsulation, the hidden data dependencies in the model, and its performance as platform for the application.

Our study has taught us that CommonKADS and PROforma are complementary: use CommonKADS as analysis methodology and use PROforma after the analysis phase for building a clinical procedure appli-

cation. We gave guidelines for mapping a CommonKADS model onto a PROforma model, since applying them helps in building the final PROforma model (which is the actual system).

As mentioned above, it seems that CommonKADS has its strengths on the domain and inference level and PROforma on the task level, at least in the case of our BloedLink study. This difference in strengths has had its implications on the mapping from CommonKADS to PROforma. It was easy to map the task structure of CommonKADS to PROforma, but the inference and domain layer raised problems. It has been a little disappointing that it was not possible to specify clear rules to map a CommonKADS inference- and domain layer to parts of a PROforma specification

Our study was surprising to us at three points. Firstly, the strong points of PROforma were related to requirements of medical reasoning, but they are required for many different domains, and therefor should be considered as shortcomings of the more general CommonKADS approach. Secondly, the weak points of PROforma are not in any way related to the special purpose nature of PROforma. Finally, it turns out that in the analysis phase for building a clinical procedure application CommonKADS was more appropriate then the special purpose methodology PROforma. Therefore CommonKADS and PROforma are complementary for building such clinical procedure applications.

References

[1] J. Angele, D. Fensel, D. Landes, and R. Studer. Developing knowledge-based systems with MIKE. *Journal of Automated Software Engineering*, 1998. To appear.

[2] C. Bauer and W. Karbach, editors. *Proceedings Second KADS User Meeting*, ZFE BT SE 21, Otto-Hahn Ring 6, D-8000 Munich 83, 17–18 February 1992. Siemens AG.

[3] B. Chandrasekaran. Generic tasks in knowledge based reasoning: High level building blocks for expert system design. *IEEE Expert*, 1(3):23–30, 1986.

[4] C. Corbridge, N.P. Major, and N.R. Shadbolt. Models exposed: An empirical study. In *Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge Based Systems*, 1995.

[5] J. Fox, N. Johns, and A. Rahmanzadeh A. PROMPT, protocols for medical prodedures and therapies, PROMPT design. Technical Report EC 4FP Project Nummer HC1043, Imperical Cancer Research Fund, April 1997.

[6] J. Fox, N. Johns, and A. Rahmanzadeh A. Protocols for medical procedures and therapies - a provisional description of the PROforma language and tools. In *Proceeding of AIME 97*. Springer 1997, 1997.

[7] I. Kluijt, J.O.M Zaat, J. van der Velden, J.Th.M. van Eijk, and F.J. Schellevis. ”voor een prikje”, general practicioners and blood tests. results of the national study on diseases and treatment in first-line care. *Huisarts en Wetenschap*, 34:67–71, 1991. (in Dutch).

[8] M. Linster. Sisyphus’91/92: Models of problem solving. *Int. J. of Human Computer Studies*, 40(3), 1994. Editorial special issue.

[9] C. Löckenhoff, D. Fensel, and R. Studer, editors. *Proceedings of the Third KADS User Meeting*, ZFE BT SE 21, Otto-Hahn Ring 6, D-8000 Munich 83, 8–9 March 1993. Siemens AG.

[10] D. Marques, G. Dallemagne, G. Kliner, J. McDermott, and D. Tung. Easy Programming: Empower ing People to Build Their own Applications”. *IEEE Expert*, pages 16–29, june 1992.

[11] M. Musen, S. Tu, A. Das, and Y. Shahar. A component-based architecture for automation of protocol-directed therapy. In *Proceedings of AIME 95, Lecture Notes 934*, pages 5–13. Springer, 1995.

[12] Den Haag: Houses of Parliament. Financial survey medical care. Parliamentary session 1993-1994, 23407, nrs 1-2, pag. 243, (in Dutch).

[13] R.J. Offen and R. Jeffery. Establishing software measurement programs. *IEEE Software*, pages 45–53, march/april 1997.

[14] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modelling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[15] A. Th. Schreiber and W. P. Birmingham. The Sisyphus-VT initiative. *International Journal of Human-Computer Studies*, 43(3/4):275–280, 1996. Editorial special issue.

[16] A. Th. Schreiber, B. J. Wielinga, J. M. Akkermans, W. Van de Velde, and A. Anjewierden. CML: The CommonKADS conceptual modelling language. In L. Steels, A. Th. Schreiber, and W. Van de Velde, editors, *A Future for Knowledge Acquisition. Proceedings of the 8th European Knowledge Acquisition Workshop EKAW'94*, volume 867 of *Lecture Notes in Artificial Intelligence*, pages 1–25, Berlin/Heidelberg, September 1994. Springer-Verlag.

[17] I. A. van Langevelde, A. W. Philipsen, and J. Treur. Formal specification of compositional architectures. In B. Neumann, editor, *Proceedings ECAI'92, Vienna*, pages 272–276, Chichester, 1992. Wiley. Longer version available as: Report IR-282, Mathematics and Computer Science, Free University of Amsterdam.

[18] M.A.M. van Wijk, A.M. Bohnen, and J. van der Lei. Advice on blood-tests in the standards of the dutch association of general practicioners: sufficiently consistent for use by the practicioner in problem-oriented request-protocols. *Ned Tijdschrift voor Klinische Chemie*, 21:285–289, 1996. (in Dutch).

[19] M.A.M van Wijk, B.M.T. Mosseveld, A.M. Bohnen, and J. van der Lei. Requests of laboratory tests: experiences with bloedlink in the delft region. In P.C.G.M. Sollet, editor, *Huisarts, Specialist en het Elektronische Medisch Dossier, Werken aan transmurale zorg*, pages 13–25. Rotterdam: Erasmus Universiteit, 1997. (in Dutch), ISBN 90-74479-06-5.

[20] B. J. Wielinga and J. A. Breuker. Models of expertise. In *Proceedings ECAI–86*, pages 306–318, 1986.

[21] B. J. Wielinga, A. Th. Schreiber, and J. A. Breuker. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1):5–53, 1992. Special issue 'The KADS approach to knowledge engineering'. Reprinted in: Buchanan, B. and Wilkins, D. editors (1992), *Readings in Knowledge Acquisition and Learning*, San Mateo, California, Morgan Kaufmann, pp. 92–116.

[22] E. Yourdon. *Modern Structured Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.