Re-engineering IMPRESS and X-MATE using CommonKADS

John Kingston

AIAI-TR-144

7 April 1994

This paper won the prize for Best Technical Paper at the BCS SGES Expert Systems '93 conference, Cambridge, 11-13 December 1993. Proceedings of the conference were published jointly by the BHR group and the SGES.

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

Abstract

The CommonKADS methodology, which is currently being developed as a successor to the KADS methodology, has suggested a considerable number of alterations or extensions to the modelling approaches used in KADS. This paper reviews these changes by applying the CommonKADS approach to two KBS systems which were originally engineered using a version of KADS: the IMPRESS system, for diagnosing faults in plastic moulding machinery, and the X-MATE system, for advising on whether a mortgage application should be granted. The focus of the work is on the modelling of expertise (domain, inference and task levels) in CommonKADS.

The conclusion of the paper is that the new approaches recommended by CommonKADS are all valuable for KADS-based KBS projects, although some of the 'libraries' of techniques are currently very small. However, the new approach to domain modelling would benefit from some guidance on ontological classification.

1 Introduction

CommonKADS is the name of the methodology which is emerging from the ongoing KADS-II project which is being funded under the ESPRIT programme. CommonKADS is an extended and revised version of the KADS methodology, which is probably the most widely used methodology for KBS development in Europe. CommonKADS also aims to incorporate parts of other knowledge engineering methods – principally the Generic Tasks approach [Chandrasekaran, 1988] and the Components of Expertise approach [Steels, 1990] – into the KADS methodology (see [Wielinga *et al*, 1992] for details).

It was decided that two KBS projects which had been originally developed with the aid of KADS, or a variant of KADS, would be re-engineered using CommonKADS in order to obtain first-hand experience of the advantages and disadvantages of CommonKADS. The projects chosen were the X-MATE project, which developed a KBS for deciding whether mortgages should be granted [Kingston, 1991], and the IMPRESS project, which produced a KBS for diagnosing faults in plastic moulding machinery [Kingston, 1992].

The KADS methodology views KBS development as a modelling process. To this end, it recommends that a number of different models are developed in the process of building a KBS:

- The *model of cooperation* helps identify tasks which a KBS could usefully perform.
- The *model of expertise* represents knowledge about how a task should be performed. This model is divided into four layers: *domain layer*, *inference structure*, *task structure* and *strategy layer*.

- KBS design is performed by *functional decomposition*, *behavioural design* and *physical design*. Each of these stages can be modelled, although KADS provided little guidance on modelling behavioural design or physical design.

For more information on KADS, see [Hickman *et al*, 1989], [Tansley & Hayball, 1992] or [Schreiber *et al*, 1993].

There are many differences between KADS and the current version of CommonKADS. However, the differences can loosely be grouped into six categories:

1. CommonKADS now offers an *organisational model*, which helps identify business problems and opportunities. In addition, the model of cooperation has now been renamed the *task model*, and aspects of the task modelling have been transferred to (and extended in) the *agent model* and the *communication model*.

2. There is now a suggested set of models to represent the domain knowledge in the model of expertise.

3. The 'generic' inference structures, which have often been considered the key to the success of KADS, have been found to require considerable customisation for almost every project carried out using KADS. CommonKADS provides *configurable* inference structures, which can be adapted to produce a better fit to the requirements of a particular project than any of KADS' 'generic' inference structures.

4. The task structure has undergone considerable change. CommonKADS requires a task *specification* to be written, which is then instantiated into a task *body* using one of a number of *problem solving methods*. For example, if the task was diagnosing faults in a car engine, the problem solving method chosen might be "generate and test". The choice of problem solving method replaces the former strategy level of knowledge, and the resulting task body is a big step towards the production of a design model.

5. There are now modelling languages available for describing KADS models: the high-level Conceptual Modelling Language (CML) and the Formal Modelling Language (ML^2).

6. Some guidance on project management is currently being developed.

This paper will describe the re-engineering of the domain, inference and task levels of expertise in IMPRESS and X-MATE (items 2,3 and 4 above). For more information on the Organisational and Task models, see [deHoog *et al*, 1993] and [deGreef & Breuker, 1992] respectively; for information on the project management aspects of CommonKADS, see [Taylor *et al*, 1992]. No detailed discussion of CML had been published at the time of writing, but a discussion of ML^2 can be found in [van Harmelen & Balder, 1992].

Much of the analysis carried out for this project was performed using KADS TOOL from ILOG. AIAI wishes to thank ILOG for its support in this project.

1.1 How to Model Using CommonKADS

In KADS, modelling of expertise was usually performed by selecting an generic inference structure from the appropriate library, modelling the domain sufficiently to instantiate the inference structure to the current application, and then proceeding with task modelling and design. CommonKADS suggests a number of approaches to modelling ([Wielinga *et al*, 1992]), including:

- bottom-up assembly of models from data;
- model assembly around a problem solving method (e.g. for a constraint satisfaction problem);
- model assembly from generic components (as in KADS);
- model specification based on top-down task decomposition;
- adapting models by knowledge differentiation (introducing new knowledge roles to circumvent computational or pragmatic constraints);
- model generation by structure mapping.

In the project described in this paper, the primary modelling method used was model assembly from generic components. However, occasional use of other approaches was found to be useful - bottom-up assembly was used in domain modelling, and a form of knowledge differentiation was used to ensure that all relevant domain categories were represented in the inference structure.

1.2 Overview of this Paper

This paper looks at the three levels of expertise modelling in CommonKADS in turn. For each level, a brief introduction is given, followed by description and results of the re-engineering of X-MATE and IMPRESS. Finally, an evaluation of the CommonKADS techniques for that level is provided.

Currently, CommonKADS' main guidance on generic components is at the inference level, and so the first step in the re-engineering process was to develop an inference structure. For this reason, the inference level of CommonKADS is described before the domain level. In practice, however, it was found that these two levels tended to be developed simultaneously, with modelling at one level helping to guide and refine the other. The task level was not developed until the other levels were complete, and so it is described last. The conclusion to the paper highlights the perceived strengths and weaknesses of expertise modelling in CommonKADS.

2 The Inference Level in CommonKADS: Configurable Inference Structures

When developing an expertise model in KADS, one of the first actions which a knowledge engineer performed was to identify the task type of the KBS application (examples of task types include heuristic classification, assessment, and configuration). On the basis of this decision, an inference structure was selected from KADS' library of task-related models[1]. The next step was to instantiate the knowledge roles and inference actions in the inference structure to terms from the domain. However, it was commonly found that this process required alterations to the structure of the generic model, rather than merely instantiating its nodes; both the X-MATE and the IMPRESS projects demonstrated this. CommonKADS' solution to this problem is to decompose the inference structures in the library into components, and to provide guidance on configuring an inference structure to a particular application. The guidance is provided by a set of questions which the knowledge engineer must ask himself about the project.

At present, the CommonKADS consortium has only defined configurable inference structure components for the Assessment task type. X-MATE's task of deciding whether to grant mortgages was identified as an assessment task, while the IMPRESS project classified the diagnosis faults in plastic moulding machinery as a systematic diagnosis task. The X-MATE project will therefore be used to provide the worked example for this section.

2.1 Using KADS on the X-MATE Project

The main contribution of KADS to the X-MATE project was the inference structure for assessment tasks. This inference structure is shown in Figure 1. When the X-MATE project was carried out, it was found that this structure needed to be changed in at least one respect in order to reflect the task of mortgage application assessment: the "ideal system model" in the top right-hand corner had to be changed to "several typical non-ideal cases". The reason for this change was that mortgage application assessment is carried out by trying to identify danger signals in mortgage applications, rather than identifying aspects of the application which match the profile of an ideal applicant. See [Kingston, 1991] for more details.

[1] This library is often known as the library of *interpretation models* because, at least in theory, its models contained more guidance than inference structures alone.

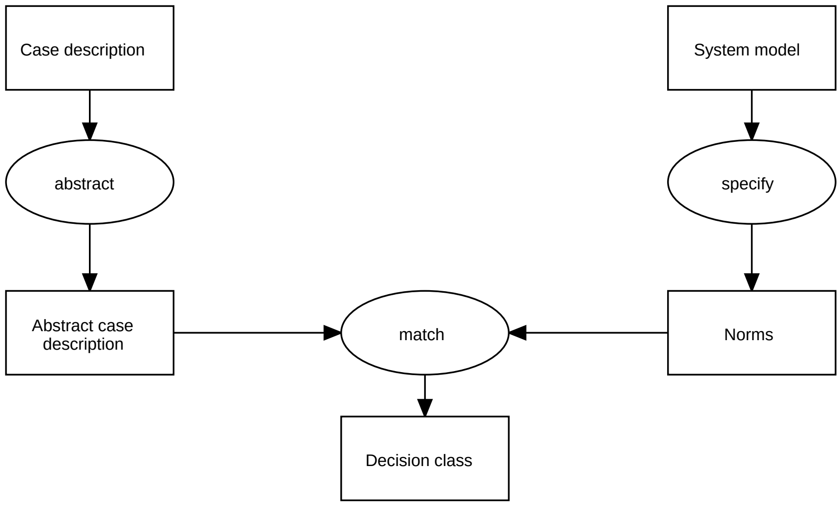

Figure 1: KADS inference structure for Assessment tasks

2.2 Using Configurable Inference Structures on the X-MATE Project

In CommonKADS, however, the basic model for Assessment tasks is simply the matching of a case description with a system model to produce a decision (Figure 2). This model is then extended by asking a series of questions about the application.[2] These questions ask the knowledge engineer about each knowledge role. Depending on the answer to each question, inference functions and knowledge roles may be inserted into the inference structure. For example, if the question:

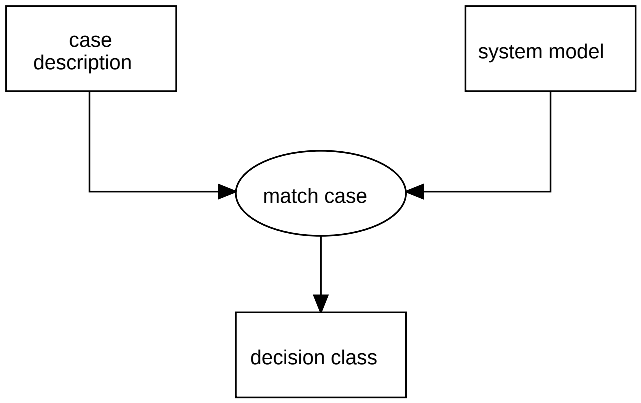

Figure 2: Basic inference structure for Assessment tasks in CommonKADS

[2]The full set of questions, and of consequent model components, can be found in [Löckenhoff & Valente, 1993].

- Is the case description already abstract enough to be matched?

was answered NO, then an **abstract** inference function and an **abstract case description** knowledge role would be added between the **case description** knowledge role and **match case** inference function.

For the X-MATE project, the questions were answered as follows:

- Is the case description already abstract enough to be matched?
  YES. A mortgage application form contains all the requisite information in an accessible form.

- Is the system model already specific enough to be matched?
  NO. As a result of this answer, a **specify** inference function is added to the inference structure, and further questions are asked about the specification process.

- Is the system model suitable for use in the specification process (or does it need to be focused because there is more than one type of system?)
  It needs to be FOCUSED since there are 3 "system models", which correspond to the 3 main reasons for defaulting on mortgages. A **focus** inference function is therefore added.

- Is the specification of the measurement system independent from the case description?
  YES. Therefore, the **case description** should provide input to the **focus** inference function, not the **specify** inference function.

- Is the decision class the direct result of matching the case against the measurement system (i.e. measuring the case)?
  NO. The decision class depends on the sum of several matches of the case against the measurement system.

- Is the decision class the result of a computation?
  YES. As a result, a **compute** inference function and another **specify** inference function are included.

The resulting inference structure is shown in Figure 3, and its instantiation to the domain of mortgage application assessment is shown in Figure 4.

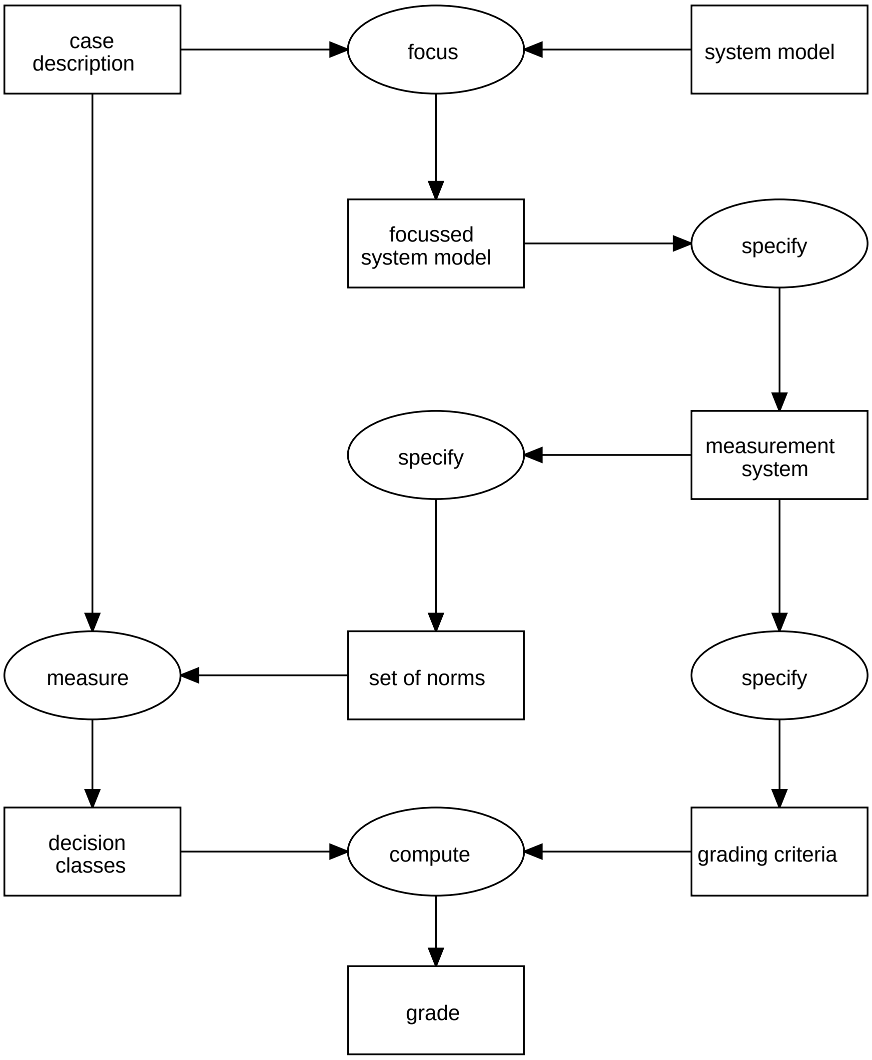

Figure 3: Inference structure for Assessment tasks, configured to the task of mortgage application assessment

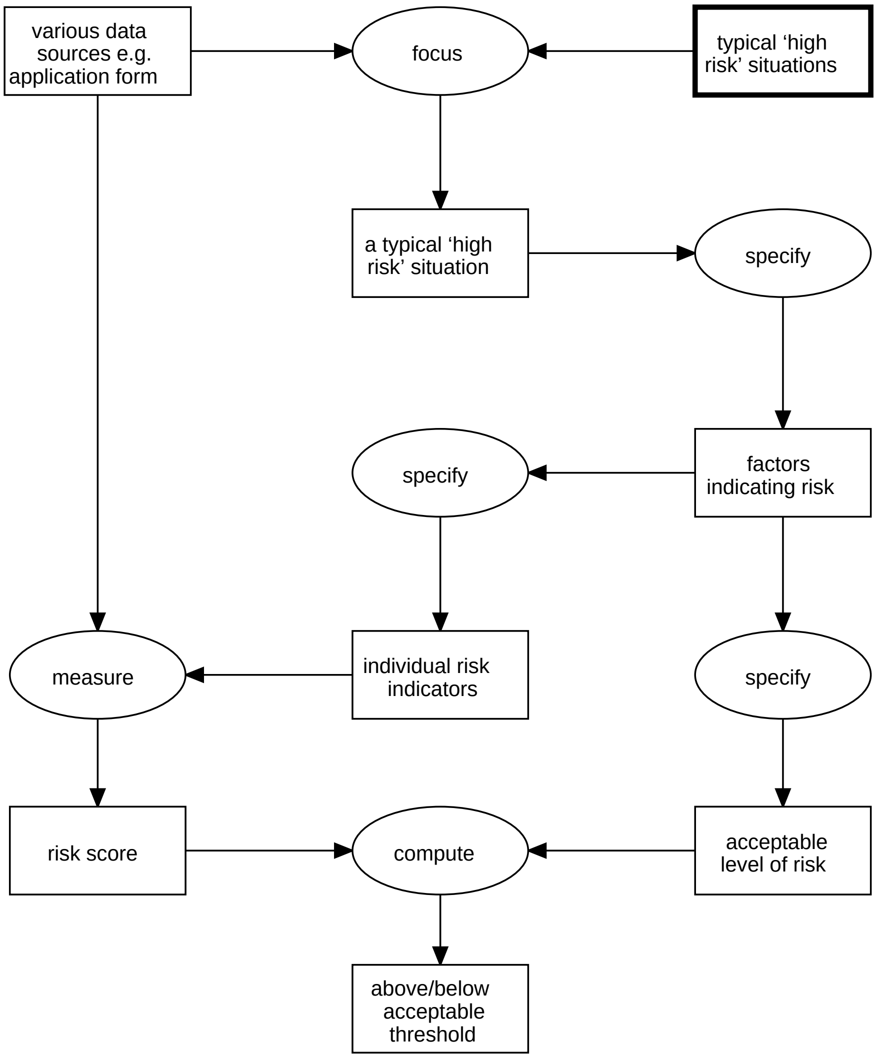

Figure 4: Configured inference structure for Assessment tasks, instantiated to the domain of mortgage application assessment.[3]

2.3 Evaluation of Configurable Inference Structures

The inference structure in Figure 3 reflects the process of mortgage application assessment much more accurately than the structure shown in Figure 1. The con-

[3]The knowledge role which is outlined in bold lines represents **static** knowledge – that is, the knowledge in this knowledge role is not changed during the inference process. The distinction between static and dynamic knowledge roles is another innovation in CommonKADS.

figuration process takes little time, and can be done even by novice knowledge engineers (see [Robertson, 1993] or [van Beuzekom, 1993]). On the basis of these observations, configurable inference structures are judged to be a valuable tool for knowledge modelling.

3 The Domain Level in CommonKADS: Domain Models

Having configured an appropriate inference structure, the domain level of the expertise model can be completed. While domain modelling was recommended in the KADS methodology, the only real guidance given was on the analysis of transcripts from interviews; it was suggested that the knowledge engineer should identify domain *concepts* from the transcript, and if possible, structure these in a hierarchy. CommonKADS has taken this idea and extended it to suggest the construction of:

- a *domain ontology*, which broadly corresponds to defining a number of dictionaries of domain terms. It is suggested that the knowledge engineer defines 'dictionaries' of
  - concepts;
  - properties;
  - relations;
  - expressions (one or more statements of the form *property = value*, which can be conjoined to produce rules).
- a number of domain models. Typically, there will be one domain model for each relation identified: for example, if the relation **causes(A,B)** has been identified then a causal domain model will be defined, which displays all the terms which are related by the **causes** relation.

In addition, CommonKADS suggests that a *model ontology* and *model schema* are defined. These represent the domain models at a more abstract level. The purpose of these models is to provide an explicit link between the domain models and the inference structure, and also to produce a representation which can be re-used in other KBS applications which perform diagnosis. The model ontology represents the domain ontology at a more abstract level (for example, the relation **subsystem-of** in the domain ontology might be represented as **part-of** in the model ontology); the model schema represents all the domain models, with one node for each domain model, using the terms defined in the model ontology.

The analyses below were carried out using KADS TOOL, which provides good support for building domain ontologies and defining domain models. The KADS TOOL output for these analyses forms the appendix to [Kingston, 1993].

3.1 Domain Modelling for the IMPRESS System

Domain ontology: A transcript from an IMPRESS knowledge elicitation session was used as the basis for the re-engineering exercise. Concepts, properties, relations and expressions were identified, created in appropriate dictionaries and linked to the transcript. KADS TOOL also supports the identification of *inferences* and *tasks* in a transcript; a number of tasks were identified in the IMPRESS transcript. A portion of the transcript, with its associated dictionaries (i.e. domain ontology), is shown in Figures 5 and 6.

> TECHNICIAN: Here's a faulty part – as you can see, the fault is *black* specks, on the back face of the moulding, on the sides of the moulding – *all over*, in fact. [**He scratches a speck with his pocket knife**]. They're quite *deeply embedded* – not *surface* specks. That means that the problem is being CAUSED by something in the material or in the process, rather than external dust, or dripping water. [He speaks to the machine operator]. *How long* has **the job been running**?

Figure 5: Part of a transcript describing diagnosis of plastic moulding machinery

Concepts (underlined)
- **concept** faulty part;
- **concept** fault;
- **concept** specks;
- **concept** contaminated material;
- **concept** process fault;
- **concept** external dust;
- **concept** dripping water;

Properties (in *italic font*)
- **property** colour of specks
  - black, etc
- **property** location of specks
  - **value-set:** all over, etc;
- **property** depth of specks
  - **value-set:** deep, surface, etc;
- **property** duration of job
  - **value-set: value-set:** 2 days, etc;

Relations (in SMALL CAPS)
- **relation** causes;

Tasks (in **bold font**)
- **task** scratch specks with pocket knife;
- **task** ask duration of job;

Figure 6: Domain ontology elicited from the transcript shown in Figure 5

Domain models: When the identification of concepts etc. in the acquired knowledge is complete, the next step is to build one or more domain models. The experience gained on this project suggests that it is wise to use the inference structure as a guide in deciding which domain models to build. The configured inference structure for the IMPRESS system (Figure 7), which is derived from the generic inference structure for systematic diagnosis tasks[4], suggests that the domain models might include the following:

- a link between complaints (symptoms) and hypothesised faults (based on the **decompose** inference function);
- a link between tests and observable properties (based on the **select** inference function);
- a link between observable properties and hypothesised faults (based on the **refine** inference function);
- a decomposition of a plastic moulding machine into its subcomponents (based on the **system model** knowledge role).

All of these suggested relationships are supported by the domain ontology:

- The link between complaints and hypothesised faults is represented by the relation **causes**;
- The link between tests and observable properties is represented by the relation **observes**;
- The link between observable properties and hypothesised faults is represented by the relation **indicates**;
- The decomposition of a plastic moulding machine into its subcomponents is represented by the relation **part of**.

Four domain models were therefore constructed to represent each of these relationships. Part of the behavioural model (which represents the **indicates** relation) is shown in Figure 8, using the semantic net representation which is usually used within Kads Tool to represent domain models.

[4]This inference structure is intended to represent the configured inference structure for the IMPRESS system. In reality, there is no guidance available for configuring inference structures for systematic diagnosis tasks, and so this model has been based on actual experience with the original IMPRESS project.

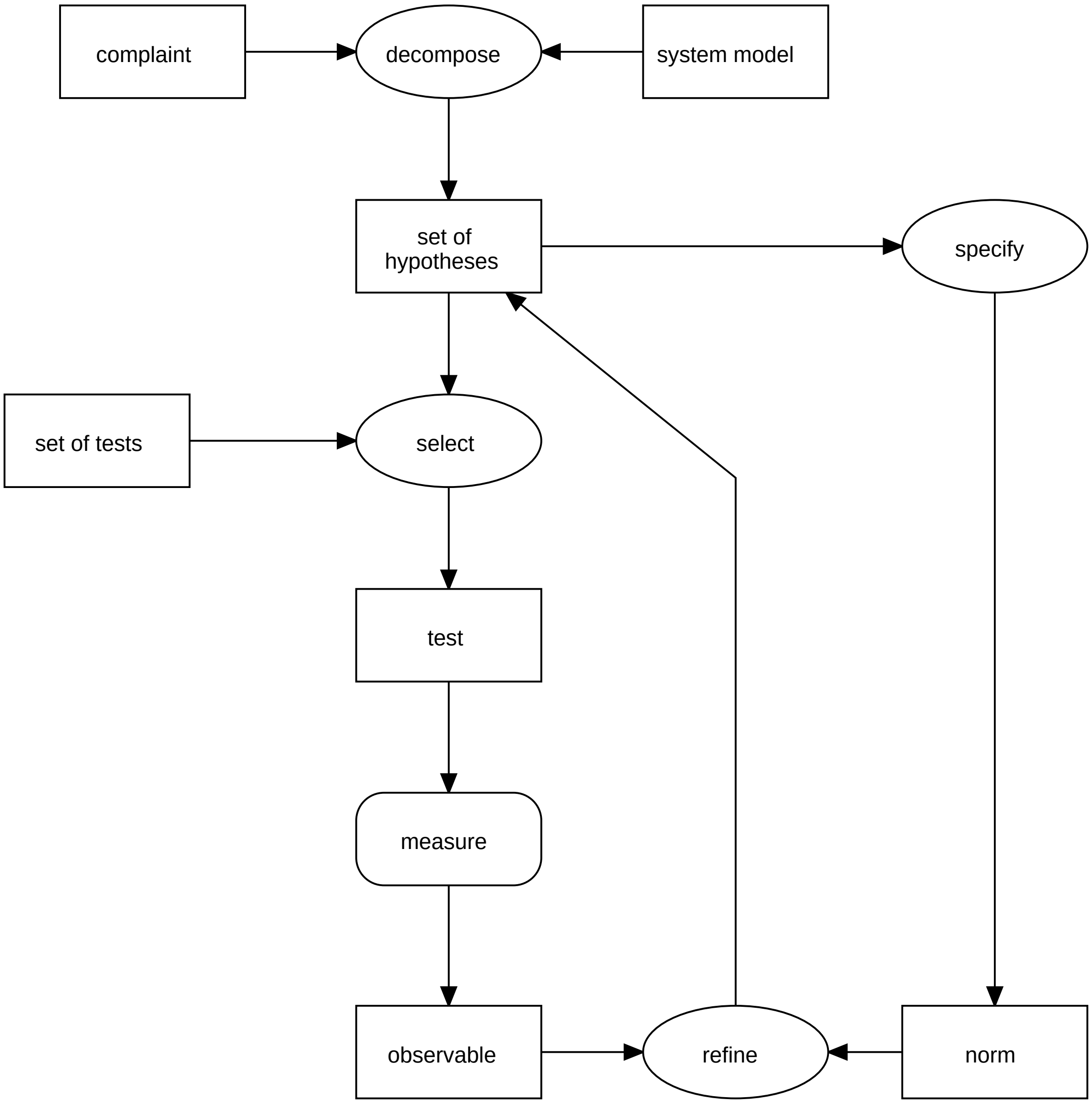

Figure 7: Configured inference structure for IMPRESS[5]

[5]The rounded rectangle around *measure* indicates that "measure" is not, strictly speaking, an inference function; instead, it is a transfer task, (see section 3.3). This syntax is another innovation in CommonKADS.

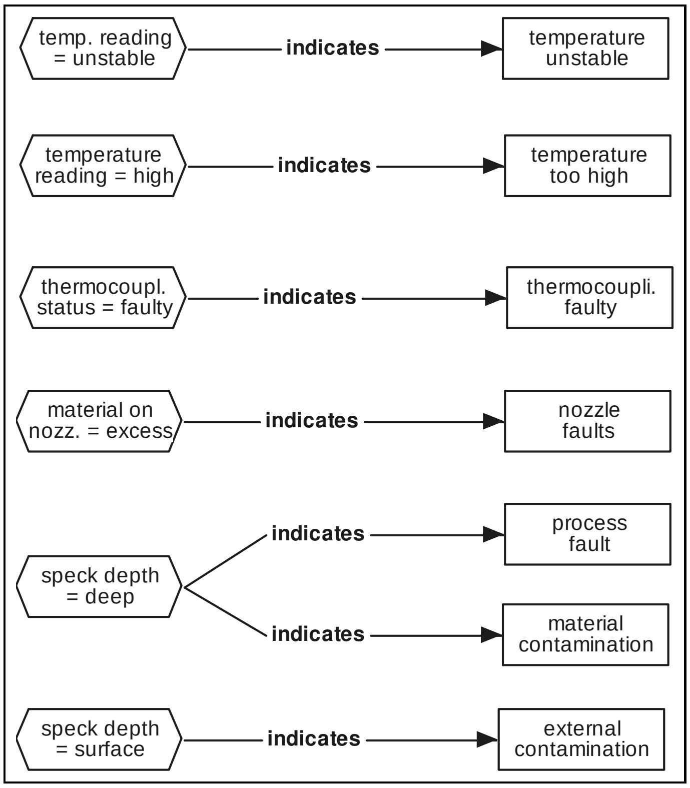

Figure 8: Part of the behavioural domain model for IMPRESS

In addition, there are some concepts which have been identified, but are not yet found in any domain model. This discovery requires a decision from the knowledge engineer: do these concepts need to be represented in a domain model, or can they safely be ignored? In the case of the IMPRESS system, the extra concepts included several concepts which referred to various states of the machine; from the transcript, it became obvious that a number of tests required the machine to be in a certain state. As a result, a new relation – **requires** – was created, and a domain model of preconditions was built to represent the requirements of tests for certain states of the machine.

Model ontology and model schema: The model schema for IMPRESS is shown in Figure 9. The terms used (i.e. the model ontology) can readily be seen to map each node to one domain model, with the exception of *manifestations*, which are defined as expressions on observable properties. The relations from the domain ontology are considered to be sufficiently abstract to be used without alteration.

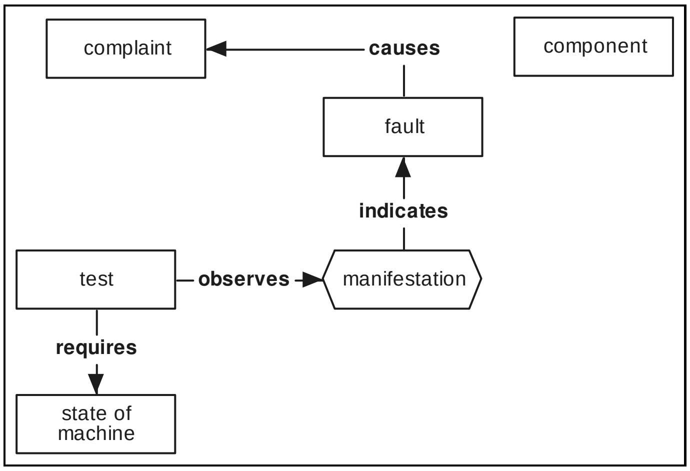

Figure 9: Model schema for IMPRESS

It can be seen from the model schema that the domain model of machine components, which represents the decomposition of a plastic moulding machine, has no links with the remainder of the domain model. This suggests that the use of an explicit decomposition of the machine is not essential for the process of diagnosis – which was actually the case in the original IMPRESS project. The model schema can therefore be used to identify concepts which do not need to be built into the final KBS.

The nodes and relations in the model schema should map directly to knowledge roles and inference functions in the inference structure. The addition of the domain model of preconditions therefore necessitated a change in the inference structure. The final top level inference structure for IMPRESS is shown in Figure 10. (The double oval for the **select** inference function indicates that this inference function is expanded into a more detailed structure at a lower level in the analysis).

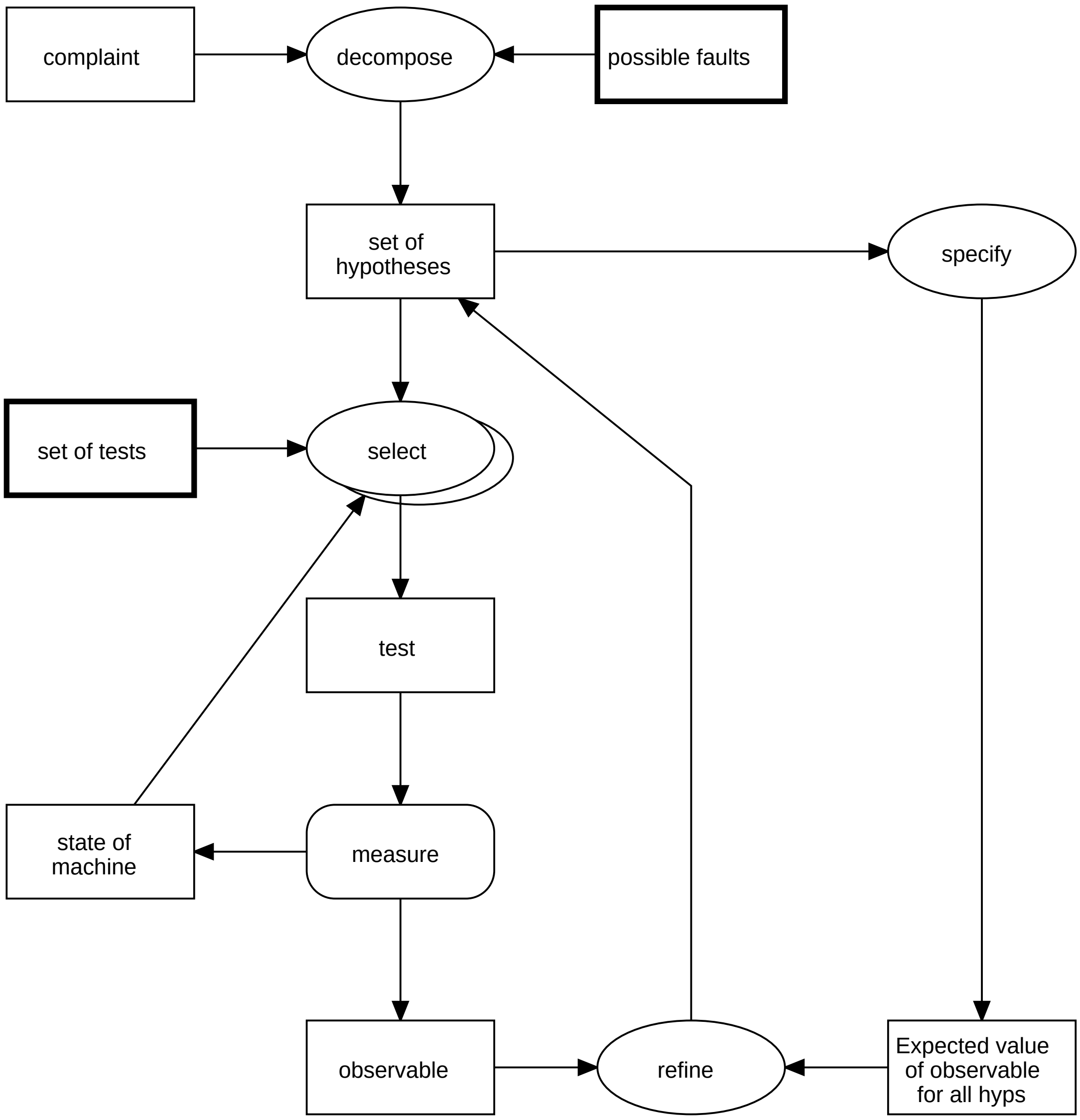

Figure 10: Final instantiated inference structure for IMPRESS

3.2 Domain Modelling for the X-MATE Project

The domain model for the X-MATE project turned out to be surprisingly simple. The domain ontology contained many rules but hardly any relations. As a result, only two domain models were identified: a hierarchy of professions (since certain categories of professions are more prone to income fluctuations than others), and a hierarchy of risks (where the top level nodes in the hierarchy represent the "typical non-ideal cases" identified in the original X-MATE project). The development of the model ontology proved useful, because it helped highlight the fact that different properties had to be acquired from different sources – a factor which was a key to the design of the X-MATE system, because the different sources required widely different amounts of effort from the user of X-MATE. This necessitated a change to the inference structure: a *select* inference action and a **selected data source** knowledge role were added between the *various data sources* knowledge role and the *measure* transfer task.

The final inference structure for X-MATE is shown in Figure 11, which follows the discussion of the task level. This is because the development of the task level for X-MATE resulted in some knowledge roles being identified as static knowledge roles rather than as dynamic knowledge roles. Discussion of this transformation can be found in section 4.2.

3.3 Evaluation of Domain Modelling

Performing the task of domain modelling was an enlightening exercise. The elicitation of a domain ontology and the subsequent construction of domain models was found to be a valuable exercise for the following reasons:

- It provided models of various aspects of the domain. This dissection of the domain, and the "cross-checking" effect of using the same concepts in more than one domain model, is an effective way of checking that all the necessary knowledge has been acquired.
- It provides a "theory" of the domain which is consistent, (hopefully) complete, and which has inter-relationships explicitly represented.
- It provides a structured approach to making alterations to an inference structure (over and above those made during the configuration process) when instantiating the inference structure to the domain.
- It provides (and possibly uses) re-usable models of the domain.

During domain modelling, however, some difficulties arose which are worthy of comment. The first difficulty was simply the sheer number of concepts which can be found in a transcript. The domain modelling exercise was carried out using one transcript of a protocol analysis session, which contained only 600 words, and yet it produced 50 concepts and 25 other items in the domain ontology. Since an average human being can speak at about 10,000 words per hour (see [Liebowitz, 1993]), the time required to identify concepts in a single transcript could potentially be very large indeed[6]. The use of KADS TOOL (as opposed to using a highlighter pen and paper) makes this task feasible, but it can still be onerous. A possible solution to this problem would be to use structured knowledge acquisition techniques such as the laddered grid, the repertory grid, or card sorting ([Diaper, 1989]), but none of these are likely to acquire *all* the necessary knowledge ([Burton *et al*, 1987]), and none are currently supported by KADS TOOL.

The second difficulty affects the process of domain modelling itself. The development of an inference structure provides guidance to a knowledge engineer on which concepts and relations can be expected in the domain, but this is not always sufficient guidance on determining the ontological type of a fragment of acquired knowledge. For example, if the transcript indicated that the complaint may be due to a fault in the thermocouplings of the plastic moulding machine, the choices for representation in the domain ontology might include:

concept thermocouplings-faulty;

concept thermocouplings
    property faulty
        value-set yes, no;

concept thermocouplings
    property status
        value-set OK, faulty;

property thermocouplings-status
    expression thermocouplings-status = faulty;

An attempt has been made to develop heuristics to help in ontological assignment (e.g. "If the item can have properties of its own, then it is a concept; if it cannot have properties of its own, it is a property"), but these heuristics have proved difficult to apply, largely because different domain models present different views on the knowledge base ([Robertson, 1993]).

For the domain modelling of IMPRESS, the inference structure provided sufficient guidance to make most ontological decisions. (For the record, the information about faulty thermocouplings was represented both as a concept, since **thermocouplings-faulty** is a fault, and also as a property, because **thermocouplings-status** can be checked by the technician). However, the ontological assignment of tests presented considerable difficulties. In the sample application provided with KADS TOOL (diagnosis of faults in a printer), tests are considered to be **transfer tasks** (tasks in which the user or another external information source transfers knowledge to the knowledge base). This is also true of the IMPRESS project –

[6] This is acceptable if all these concepts are to be used in the final KBS, but at this stage of the modelling process bottom-up assembly is being employed, and so it is not known which concepts will ultimately be used.

tests obtain data about the plastic moulding machine, and this data is reported to the KBS. However, CommonKADS does not allow tasks to have properties – and yet a key part of the reasoning in IMPRESS is to decide which test to perform next, on the basis of the time required for that test and the explanatory power of the properties which are measured by the test. In order to represent this, tests had to be described both as tasks **and** as concepts, which is far from ideal.

A final difficulty is that domain modelling is only intended to represent semantic relationships, hence the use of semantic nets in KADS TOOL to represent domain models. Semantic networks only allow a given node to appear once in any one model. This is a problem when using CommonKADS to model non-semantic relationships. An example can be found in [van Beuzekom, 1993], where modelling of molecular structures using KADS TOOL proved difficult because organic molecules may contain many carbon atoms, and KADS TOOL insisted that each atom was represented using a different concept.

On balance, the construction of domain models and a model schema is deemed to be a useful activity when constructing a KBS. The knowledge engineer should, however, be aware of the potential difficulties.

4 The Task Level in CommonKADS: Problem Solving Methods

The third level of expertise modelling in CommonKADS is the task level. CommonKADS requires a task *definition* to be written, which is then instantiated into a task *body* using one of a number of *problem solving methods*. The task specification can be derived from the inference structure (or rather, from the CML description of the inference functions – see the appendix of [Kingston, 1993] for a worked example); the major decision at this stage is which problem solving method to use.

Problem solving methods are a prescription of the way in which a certain class of task definitions can be satisfied. They specify the relation between a task definition and a task body, by mapping the task specific terms on to the (generic) terms used in the method description [Wielinga *et al*, 1992]. It follows that the choice of the most appropriate problem solving method is made by comparing the task definition with the method description of each problem solving method.

4.1 Choosing a Problem Solving Method for IMPRESS

The top level task definition for the IMPRESS system is given below, using CML and first order predicate logic. (Note that capital letters such as H and M are used to represent sets of data.)

Task definition for IMPRESS:

task machine-fault-diagnosis(c, f)

goal: Find a fault f that explains a given symptom c
$\wedge$ all manifestations observed indicate f
$\wedge$ no other fault is indicated by all the observed manifestations

roles:
case-initial-input: c: complaint
case-user-input: M: set of manifestations
solution: f: fault

task-specification:
covers(f, c)
$\wedge$ ($\forall$ m:manifestation indicates(m,f))
$\wedge$ $\neg$($\exists$ f2:fault $\wedge$ covers(f2,c)
$\wedge$ ($\forall$ M:manifestation indicates(m,f2)))
$\vdash$ solution(f);
covers(f, c)
$\wedge$ ($\forall$ m:manifestation indicates(m,f))
$\wedge$ ($\exists$ f2:fault $\wedge$ covers(f2,c)
$\wedge$ ($\forall$ m:manifestation indicates(m,f2)))
$\wedge$ ($\exists$ t:test observes(t,m) $\wedge$ indicates(m,f))
$\vdash$ perform(t);

The task specification states that fault f is a solution if:

- f covers (i.e is capable of causing) the observed complaint;
- all the manifestations (observed properties) indicate that f could be true;
- there is no other fault for which the above two conditions are true.

However, if there are still two or more faults under suspicion, the task specification states that a test should be performed to investigate one of those faults.

The knowledge engineer's task now is to choose a problem solving method. In this example, the choice has been narrowed down to two options: *generate and test* or *confirmation by exclusion*. The method definitions for these are given below.

Problem solving method: generate and test:

problem solving method *generate and test*
goal: G: FIND(s:solution)

task-characterisation:
criterion1(s) $\wedge$ criterion2(s) $\vdash$ solution(s)

control-roles:

c: complaint
h: hypothesis $\rightarrow$ solution
sub-tasks:
generate(complaint, hypothesis)
test(hypothesis)
method-definition:
A1: $\forall$ x solution(x) $\vdash$ generate(x)
A2: $\forall$ x generate(x) $\wedge$ test(x) $\vdash$ solution(x)
A3: $\forall$ x generate(x) $\vdash$ criterion1(x)
A4: $\forall$ x test(x) $\vdash$ criterion2(x)
A1 $\wedge$ A2 $\wedge$ A3 $\wedge$ A4 $\vdash$<P1> $\exists$ s solution(s)
task-expression-schema P1
REPEAT
generate(c,h)
UNTIL test(h)
RESULT(h)

Problem solving method: confirmation by exclusion:

problem solving method *confirmation by exclusion*
goal: G: FIND(s:solution)
task-characterisation:
criterion1(s) $\wedge$ $\neg\exists$ criterion2(s) $\vdash$ solution(s)
control-roles:
c: complaint
h: hypothesis $\rightarrow$ solution
H: set of hypotheses
M: set of manifestations
n: number of hypotheses in H
sub-tasks:
generate(complaint, set of hypotheses)
test(hypothesis)
refine(set of hypotheses)
compute(number of hypotheses in set of hypotheses)
method-definition:
A1: $\forall$ x solution(x) $\vdash$ generate(x)
A2: $\forall$ x generate(x) $\wedge$ set(manifestations) $\vdash$ solution(x)
A3: $\forall$ x generate(x) $\wedge$ set(manifestations) $\vdash$ criterion1(x)
A4: $\forall$ x generate(x) $\wedge$ set(manifestations) $\vdash$ criterion1(x)
A1 $\wedge$ A2 $\wedge$ A3 $\wedge$ A4 $\vdash$<P2> $\exists$ s solution(s)
task-expression-schema P2
generate(c,H)

REPEAT
  test(h) → M
  refine(M, H)
UNTIL
n <= 1

IMPRESS task body: It is clear that these two method definitions are very similar. Both can be applied if the task specification can be interpreted as a conjunction of two criteria, and both involve generating and repeatedly testing hypotheses. However, the task characterisation of the method for confirmation by exclusion indicates that the method is dependent on the **non-existence** of the second criterion, which is a key feature of the task specification. Further examination of the task reveals that it fulfils all the statements of the method definition, and so confirmation by exclusion is chosen as the problem solving method for IMPRESS. The resulting task body is as follows:

IMPRESS task body:

task body

sub-goals:

G1: FIND all fault states h with covers(h,c)
G2: TEST a manifestation m such that $h \in H$
   $\wedge$ indicates(m, h)
G3: REFINE the set of hypotheses by removing all h for which indicates($\neg$m, h)

sub-tasks:

G1: generate(c,H)
G2: test(h → m)
G3: refine(m, H)

control-roles:

hypothesis h: fault
manifestation m: manifestation
number of hypotheses in H n: positive integer

task-expression

generate(c,H)
REPEAT
  test(h) → m
  refine(m, H)
UNTIL
n <= 1

4.2 Choosing a Problem Solving Method for X-MATE

The task modelling for X-MATE produced a very simple task structure. The reason for this is that much of the knowledge required for mortgage application assessment – the specification of a measurement system, the specification of risk indicators, and so on – has been **compiled** into a set of rules. In CommonKADS terminology, much of the inference has been done in advance, producing knowledge roles which are now static knowledge roles, from the viewpoint of the KBS. Figure 11 shows an inference structure which indicates the processing which is actually performed by X-MATE. The obvious problem solving method for such a problem is to use rule-based pattern matching. CommonKADS does not yet provide any guidance on choosing an appropriate rule-based paradigm (e.g. forward chaining vs backward chaining); some heuristic guidance can be found in the "probing questions" approach of [Kline & Dolins, 1989] (see also [Kingston, 1992] and [MacNee, 1992]).

4.3 Evaluation of Task Modelling

It can be seen that the task body for IMPRESS shown above provides a much more detailed prescription for the design phase of CommonKADS than the task structure in the KADS methodology. The use of problem solving methods is therefore recommended. The main difficulty is that the library of problem solving methods currently contains just one method (generate and test, specified in [Wielinga *et al*, 1992]) – the method for confirmation by exclusion was defined in the course of this project, thus doubling the current size of the library! It is therefore unsurprising that little is known about techniques for choosing between similar problem solving methods. It is hoped that such techniques will be developed in due course. (For guidance on development on problem solving methods, and a theoretical underpinning of them, see [Akkermans *et al*, 1993]).

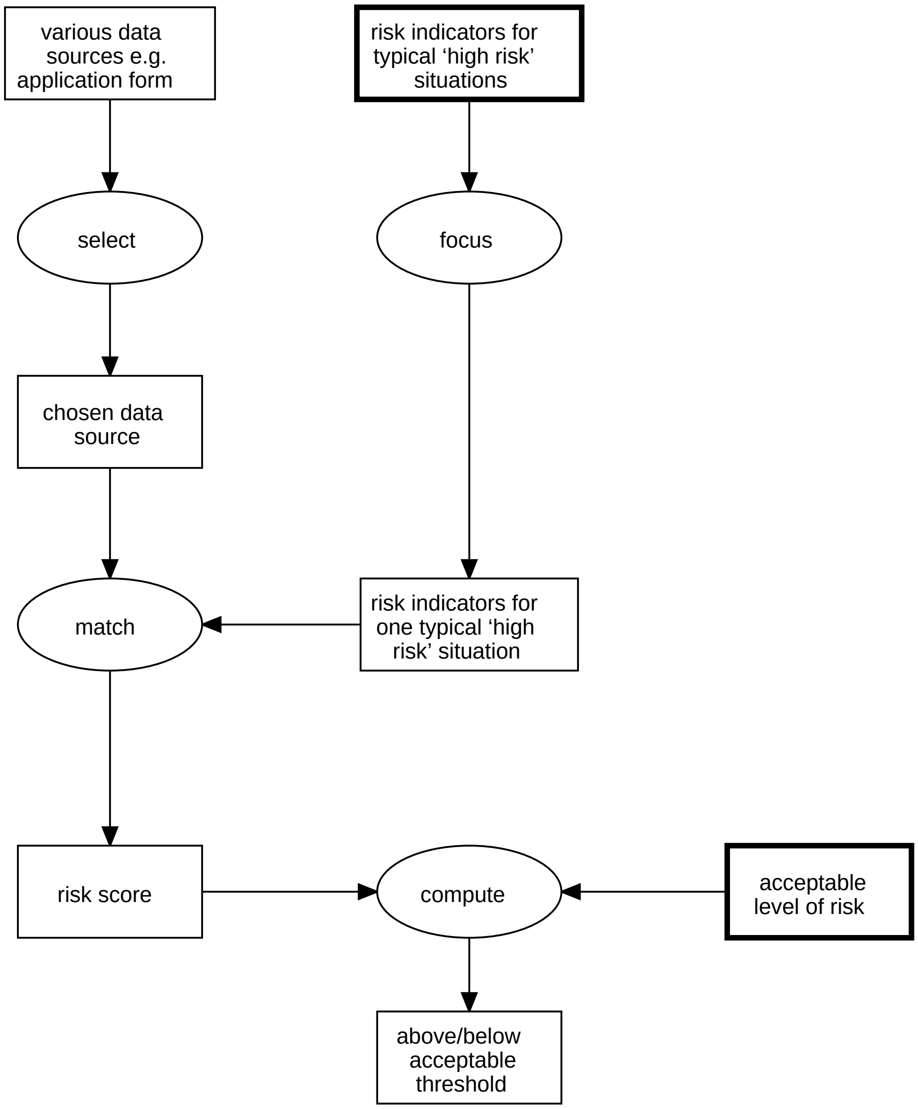

Figure 11: The actual inferences performed by X-MATE

5 Conclusion

The process of re-engineering two existing KBS applications into CommonKADS has shown that the refinements introduced to expertise modelling by CommonKADS are all useful techniques, and are recommended for future KBS projects. However, the guidance on configuring inference structures and the library of problem solving methods are currently very sparse, and need to be expanded greatly for CommonKADS techniques to be widely usable. Domain modelling in CommonKADS has undergone the greatest transformation of all from KADS; it encourages greater understanding of the domain, provides explicit links with (and justification for adaptations to) the inference level, and aids in the development of re-usable domain models. Domain modelling can be a big task, however; it would be made easier by the provision of some guidance on ontological classification.

Acknowledgements

I would like to thank Gail Anderson and Ian Filby of AIAI for their input and comments on this paper.

References

[Akkermans *et al*, 1993] Akkermans, H., Wielinga, B. and Schreiber, G. (1993). Steps in constructing problem solving methods. In *Proc. of the European Knowledge Acquisition Workshop (EKAW'93).* Also available as chapter 5 of the CommonKADS Expertise Model Definition Document, ed. Wielinga *et al*, June 1993, available by FTP from swi.psy.uva.nl.

[Burton *et al*, 1987] Burton, A.M., Shadbolt, N.R., Hedgecock, A.P. and Rugg, G. (1987). A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1. In Moralee, S., (ed.), *Research and Development in Expert Systems IV.* Cambridge University Press.

[Chandrasekaran, 1988] Chandrasekaran, B. (1988). Generic tasks as building blocks for knowledge based systems: The diagnosis and routine design examples. *The Knowledge Engineering Review*, 3(3):183–210.

[deGreef & Breuker, 1992] de Greef, H.P. and Breuker, J. (March 1992). Analysing system-user cooperation in KADS. *Knowledge Acquisition*, 4(1):89–108.

[deHoog *et al*, 1993] de Hoog, R., Benus, B., Metselaar, C. *et al.* (Jan 1993 1993). Applying the CommonKADS organisational model. Restricted circulation KADS-II/T1.1/UvA/RR/004/4.1, ESPRIT project P5248 KADS-II.

[Diaper, 1989] Diaper, D., (ed.). (1989). *Knowledge Elicitation: Principles, Techniques and Applications*. Ellis Horwood, Another very good book, surveying many useful techniques. Includes a section on using task models (such as those provided by KADS) for knowledge elicitation.

[Hickman *et al*, 1989] Hickman, F., Killin, J., Land, L. *et al.* (1989). *Analysis for knowledge-based systems: A practical introduction to the KADS methodology*. Ellis Horwood, Chichester.

[Kingston, 1991] Kingston, J. (1991). X-MATE: Creating an interpretation model for credit risk assessment. In *Expert Systems 91*. British Computer Society, Cambridge University Press. Also available from AIAI as AIAI-TR-98.

[Kingston, 1992] Kingston, J. (1992). KBS Methodology as a framework for Co-operative Working. In *Research and Development in Expert Systems IX*. British Computer Society, Cambridge University Press. Also available from AIAI as AIAI-TR-130.

[Kingston, 1993] Kingston, J. (1993). Re-engineering IMPRESS and X-MATE using CommonKADS. AIAI Technical Report AIAI-TR-130, AIAI, This report consists of the paper of the same name presented at Expert Systems '93 with an appendix showing the full analysis of the expertise model of the IMPRESS system which was carried out using KADS TOOL.

[Kline & Dolins, 1989] Kline, P.J. and Dolins, S.B. (1989). *Designing expert systems : a guide to selecting implementation techniques*. Wiley.

[Liebowitz, 1993] Liebowitz, J. (June 1993). An Interactive Multimedia Tool for Learning Knowledge Acquisition Methods. In P. Chung, G. Lovegrove and Ali, M., (eds.), *Proceedings of the 6th International Conference on Industrial and Engineering Applications of AI and Expert Systems*, pages 184–187, City Chambers, Edinburgh. Gordon and Breach.

[Löckenhoff & Valente, 1993] Löckenhoff, C. and Valente, A. (March 1993). A Library of Assessment Modelling Components. In *Proceedings of 3rd European KADS User Group Meeting*, pages 289–303. Siemens, Munich.

[MacNee, 1992] MacNee, C. (Sept 1992). PDQ: A knowledge-based system to help knowledge-based system designers to select knowledge representation and inference techniques. Unpublished M.Sc. thesis, Dept of Artificial Intelligence, University of Edinburgh.

[Robertson, 1993] Robertson, S. (Sept 1993). A KBS to advise on selection of KBS tools. Unpublished M.Sc. thesis, Dept of Artificial Intelligence, University of Edinburgh.

[Schreiber *et al*, 1993] Schreiber, A. Th., Wielinga, B. J. and Breuker, J. A., (eds.). (1993). *KADS: A Principled Approach to Knowledge-Based System Development.* Academic Press, London.

[Steels, 1990] Steels, L. (1990). Components of Expertise. *AI Magazine.* Also available as: AI Memo 88-16, AI Lab, Free University of Brussels.

[Tansley & Hayball, 1992] Tansley, D.S.W. and Hayball, C.C. (1992). *Knowledge-Based Systems Analysis and Design: A KADS Developers Handbook.* Prentice Hall, This book is part of the BCS Practitioner series.

[Taylor *et al*, 1992] Taylor, R., Bright, C., Menezes, W. and Groth, J. (1992). Life-cycle model (release 1, volume 1): Principles of the LCM. ESPRIT Project P5248 KADS-II KADS-II/T2.1/TR/TRMC/010/1.0, Deliverable D2.2a, Touche Ross & Co, Lloyds Register and Cap Programator.

[van Beuzekom, 1993] van Beuzekom, A. (1993). Analysing Molecular Similarity using a KBS and CommonKADS. AIAI Visitor Project Report AIAI-PR-56, AIAI, University of Edinburgh, The author is employed at Unilever Research Laboratories, Vlaardingen, The Netherlands.

[van Harmelen & Balder, 1992] van Harmelen, F. and Balder, J. R. (1992). $(ML)^2$: a formal language for KADS models of expertise. *Knowledge Acquisition*, 4(1). Special issue on 'The KADS approach to knowledge engineering'.

[Wielinga *et al*, 1992] Wielinga, B., Van de Velde, W., Schreiber, G. and Akkermans, H. (1992). The KADS Knowledge Modelling Approach. In *Proceedings of the Japanese Knowledge Acquisition Workshop (JKAW'92).* Also available by FTP from swi.psy.uva.nl.