Evaluation of workbenches which support the CommonKADS methodology [1]

John K.C. Kingston, Jim G. Doheny and Ian M. Filby

AIAI-TR-201

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

[1] Also appears in *Knowledge Engineering Review, 10, 3, 1995.*

Abstract

The KADS methodology and its successor, CommonKADS, have gained a reputation for being useful approaches to building knowledge based systems in a manner which is both systematic and well documented. However, these methods require considerable effort to use them completely. It has been suggested that automated support for KADS or CommonKADS users, in the form of "knowledge engineering workbenches", could be very useful. These tools would provide computerised assistance to knowledge engineers in organising and representing knowledge, in a similar fashion to the support which CASE tools provide for software engineers. In order to provide support for KADS or CommonKADS, the workbenches should provide specific support for the modelling techniques recommended by these methods, which are very detailed in the representation and analysis stages of knowledge engineering. A good knowledge engineering workbench should also be easy to use, should be robust and reliable, and should generate output in a presentable format.

This paper reports on an evaluation of two commercially available workbenches for supporting the KADS approach: KADS Tool from ILOG and Open KADS Tool from Bull. This evaluation was carried out by AIAI as part of the CATALYST project, funded by the European Community's ESSI programme, which aimed to introduce CommonKADS to two technology-oriented companies. Information is also presented on two other workbenches: the CommonKADS workbench (which will soon become commercially available) and the VITAL workbench. The results show various strengths and weaknesses in each tool.

1 Introduction

The KADS methodology [Schreiber *et al*, 1993] [Tansley & Hayball, 1993] and its successor, CommonKADS [Wielinga *et al*, 1992] [Breuker & van de Velde, 1994] are collections of structured methods for building knowledge based systems, analogous to methods such as SSADM for software engineering. These methods were developed between 1983 and 1994 on two projects funded by the European Community's ESPRIT programme. The KADS approach[2] views the development of a knowledge based system as a modelling activity, and so the heart of these methods is the construction of a number of models which represent different views on problem solving behaviour. For example, CommonKADS recommends the creation of six models. Four of these six models focus on the organisational and application context within which a knowledge based system will operate; one model represents the knowledge base of the system; and one model describes how the knowledge will be implemented within a computer program. The construction of a

[2]The terms **KADS** and **the KADS approach** are used in this paper to describe the approach taken by both KADS and CommonKADS. Where the two methods differ, this paper refers to them as **KADS-I** and **CommonKADS** respectively. The two ESPRIT projects are referred as the **KADS-I project** and the **KADS-II project**.

knowledge based system is centred around the progressive (and possibly iterative) development of these models. The models are intended to support step by step transformation of knowledge; starting from the identification of a problem or opportunity, knowledge is identified, analysed, and extended until it is possible to produce a detailed specification for an implemented system.

The creation and maintenance of all these models has been found to be an onerous task. It has been suggested that automated support for KADS or CommonKADS users, in the form of "knowledge engineering workbenches", might provide assistance in representing models and ensuring consistency between models. These workbenches could provide computerised assistance to knowledge engineers in organising and representing knowledge, in a similar fashion to the support which CASE tools provide for software engineers. Specifically, a workbench should provide facilities to create or modify KADS models quickly, and should also keep track of links between models; most models are produced by transforming the contents of another model into a different formalism, and so it is important to keep a record of these transformations to ease maintenance of the models. In addition, a knowledge engineering workbench for KADS users should provide some support for knowledge acquisition, and possibly for project management as well, since these activities are necessary for any knowledge engineering project. Ideally, the workbench should provide support for implementing the recommendations on project management and the few suggestions for knowledge acquisition which are made by CommonKADS.

The first workbench for supporting KADS users was Shelley [Anjewierden *et al*, 1990], which provided some support for KADS-I users. However, commercially supported workbenches have only appeared on the market in the last year or two. This paper describes an evaluation of two of these commercially supported workbenches: KADS Tool® (from Ilog Inc.) and Open KADS Tool® (from Bull). This evaluation was carried out by AIAI as part of the CATALYST project, funded by the European Community's ESSI programme, which aimed to introduce CommonKADS to two technology-oriented companies. The evaluation formed part of a planned programme aimed to introduce CommonKADS into companies with previous experience in knowledge engineering. The companies involved received training and consultancy in the use of CommonKADS; they were also shown, and given the opportunity to use, both KADS Tool and Open KADS Tool. An evaluation report was then prepared which gave a detailed comparison of these two tools, to give the companies a basis for deciding which tool to purchase. The evaluation was therefore focused on the needs of two commercial companies who were beginning to use CommonKADS. The evaluation was performed against a number of different criteria,

assessing both the degree of support for the CommonKADS methodology provided by each workbench, and the usability of the workbench.

This paper describes and extends the results of this evaluation, by providing information on the facilities of two other workbenches in addition to KADS Tool and Open KADS Tool. These other workbenches are the CommonKADS Workbench, which is expected to become commercially available in 1995, and the VITAL Workbench. The VITAL workbench was developed as part of the VITAL ESPRIT project [Shadbolt *et al*, 1993], and is expected to be used by the partners on the project for in-house services. The VITAL workbench is described because of the similarities between the VITAL methodology and the CommonKADS methodology [Shadbolt *et al*, 1993] [Motta *et al*, 1995], and because it provides an alternative perspective on the facilities which knowledge engineering workbenches might offer. This paper starts by describing the CommonKADS model set in section 2; it then provides an overview of each workbench in section 3, before reporting the results of the evaluation in sections 4 and 5. An extensive appendix provides a detailed summary of the features of KADS Tool, Open KADS Tool and the CommonKADS Workbench.

2 The CommonKADS Model Set

CommonKADS recommends the construction of six models [deHoog *et al*, 1993b]:

- The *organisational* model represents the processes, structure and resources within an organisation;
- The *task* model shows the tasks carried out in the course of a particular process;
- The *agent* model represents the capabilities required of the agents who perform a process, and constraints on their performance;
- The *communication* model shows the communication required between agents during a process;
- The *expertise* model is a model of the expertise required to perform a particular task. This model is divided into three components:
  - declarative knowledge about the domain;
  - the inference processes required during problem solving;

– a hierarchical classification and ordering of the inference processes.

- The *design* model culminates in the design of a knowledge based system to perform all or part of the process under consideration.

The remainder of this section describes the CommonKADS models in more detail, and gives examples of how they are represented.

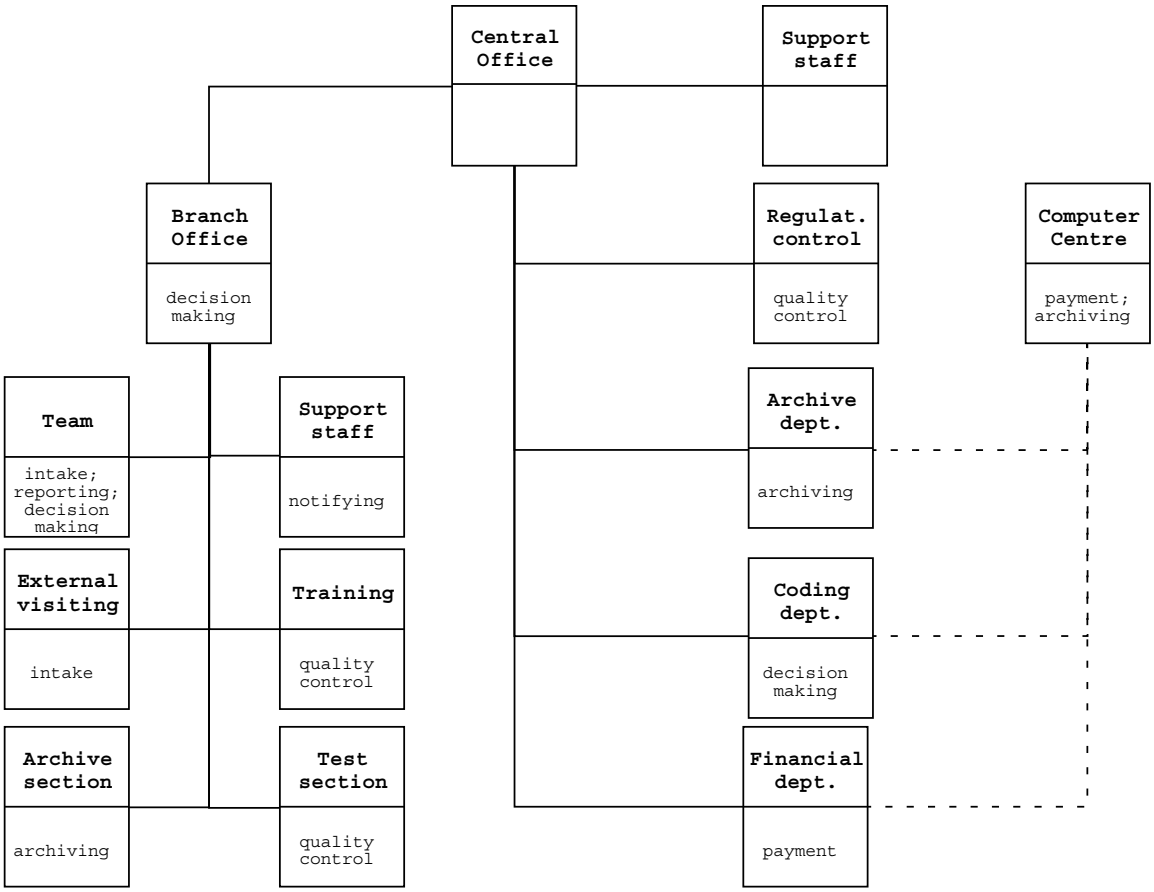

Figure 1: Organisational model: cross-product between function and structure perspectives in a Social Security department

2.1 Organisational model

The organisational model in CommonKADS examines an organisation from five major perspectives. The perspectives are:

- the **functions** of the organisation;
- the **structure** of the organisation;
- the **processes** in the organisation;
- the **power/authority** in the organisation;
- the **resources** in the organisation.

Each perspective is represented by a single diagram, or by text. These five perspectives can be combined into cross-products which provide the most useful information: for example, a project on a Social Security department produced a cross-product of function and structure which indicated that one of the functions (archiving) was being performed by three different divisions of the department [deHoog *et al*, 1993a]. A diagram representing this "cross-product" is shown in Figure 1.

The main purpose of this diagram (and subsequent diagrams in this section) is to show the graphic facilities required for representing this CommonKADS model. The diagram above shows a number of structural units (i.e. departments) within an organisation, which are represented as boxes. Each structural unit must be able to represent the functions which that unit performs; in this diagram, the functions are listed in the bottom half of each box. The structure perspective also shows communication links, which are represented here by lines between different structural units; bold lines represent inter-organisational links, and dashed lines represent intra-organisational links.

2.2 Task model

If the organisational model indicates a particular function which might usefully be automated, a task model can be produced which provides a detailed description of the tasks which carry out that function. The task model may also identify the inputs and outputs of each process (or task), and the decomposition of tasks into more specific sub-tasks. Its purpose is to allow identification of tasks which could usefully be performed by an automated system, or by a program and a user working in conjunction.

A typical task model is shown in Figure 2, using the graphical format which was originally specified in [deGreef & Breuker, 1992]. The model represents the tasks involved in the preparation of a meal; such a model might be used by a large restaurant or a hotel which was considering a reorganisation of its business.

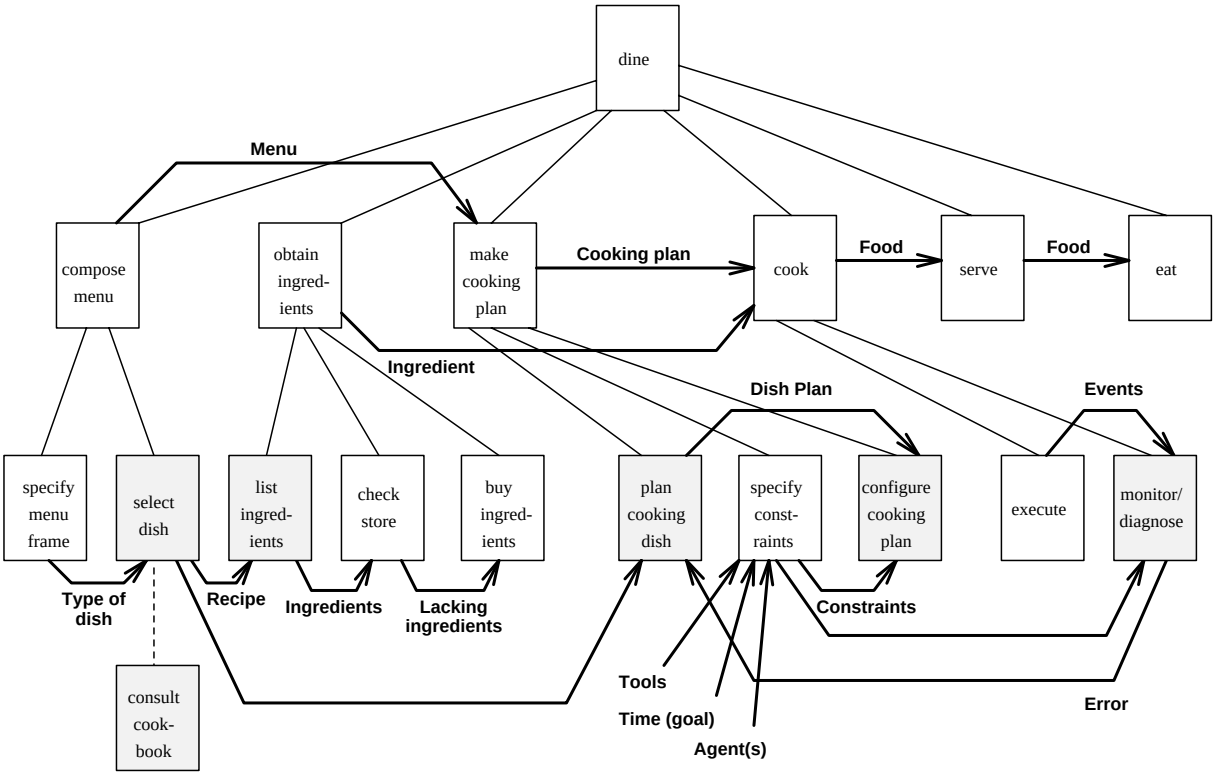

Figure 2: Task model: preparing a meal

In this diagram, the boxes represent tasks; the arrows represent inputs and outputs of tasks; the lighter lines show the decomposition hierarchy of tasks; and the shading indicates those tasks which are worth considering for implementation using a computer system.

2.3 Agent Model

The agent model represents all the agents which participate in a problem solving process. It is often useful to develop two agent models: one which is based on the task model, showing which agents are currently involved in performing particular tasks, and one which predicts the agents and capabilities required to carry out future tasks.

An example of an agent model, using the graphical format suggested by the CommonKADS Workbench, is shown in Figure 3.

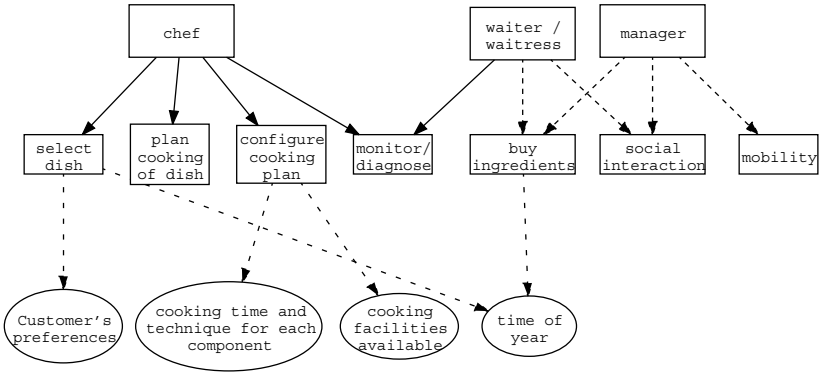

Figure 3: Agent model: Preparing a Meal

2.4 Communication Model

The communication model indicates all the transactions which take place between different agents. Normally, this will comprise transactions between a knowledge based system and other agents. It is therefore often convenient to combine the Communication model with the Agent model.

An example of an communication model, using the graphical format suggested in the KADS-II report on the CommonKADS Communication Model ([Waern *et al*, 1994]) is shown in Figure 4.

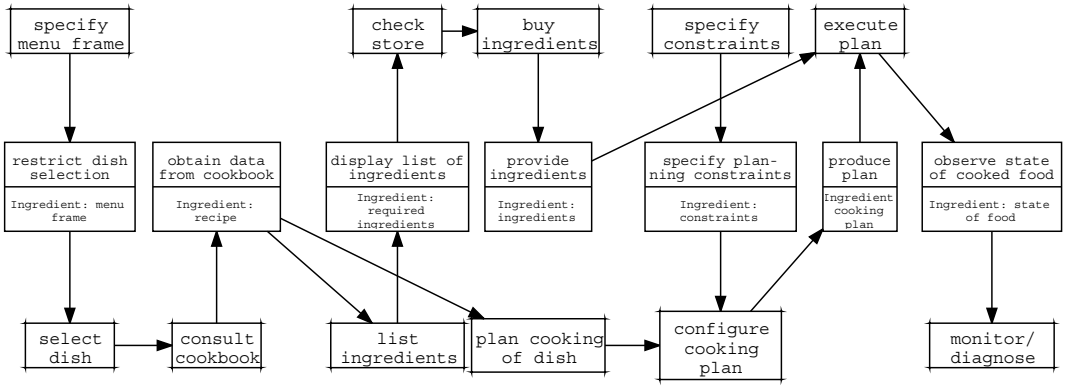

Figure 4: Communication model: Preparing a Meal

2.5 Expertise Model

As stated above, the Expertise model is divided into three components, which represent declarative knowledge, inference (procedural) knowledge, and task (control) knowledge. KADS-I also recommended a fourth subdivision (strategic knowledge). Each of these subdivisions might contain several models; the Expertise Model is therefore not a single model, but a collection of several models.

2.5.1 Expertise Model: Domain Level

The domain knowledge in the model of expertise represents the declarative knowledge which has been acquired. CommonKADS suggests that each item of declarative knowledge is classified into one of the four categories outlined below:

- **Concepts**: classes of objects in the real or mental world of the domain studied, representing physical objects or states;
- **Properties**: attributes of concepts e.g. *colour*.[3]
- **Expressions**: statements of the form "the *property* of *concept* is *value*";
- **Relations**: links between any two items of domain knowledge.

[3]CommonKADS also supports *attributes*, which are properties which can belong to multiple concepts, and which are defined independently from concepts for convenience. *Weight* might be an example of a CommonKADS attribute.

Once items of domain knowledge have been classified, they can be used in *domain models*, which show relations between different items of knowledge. For example, a domain model might show all acquired examples of one concept **causing** another; or it might show a taxonomic hierarchy of concepts, connected to each other by **is-a** relations. Figure 5 shows an example of a domain model; this model represents indicative relationships between certain symptoms of a manufacturing machine, and possible faults with that machine.

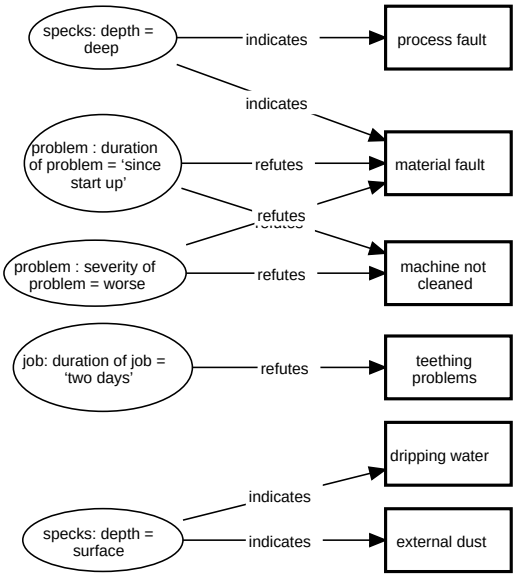

Figure 5: A domain model which links expressions with concepts

CommonKADS also recommends that the domain knowledge is represented at a more abstract level, at which generic statements about the structure and contents of domain models are expressed. The terms used at this abstract level are stored in a *model ontology*, and the statements at this abstract level are known as *model schemata*. This abstraction is intended to form a basis for re-using domain models in different problem solving situations.

The domain knowledge in the CommonKADS Expertise model therefore consists of a *domain ontology* (i.e. a collection of classified terms) and a number of *domain models* which represent relations between items from the domain ontology. In addition, it is recommended that an abstract view on the domain knowledge is created.

2.5.2 Expertise Model: Inference Knowledge

The second subdivision of the Expertise Model records knowledge about inference which has to be performed in order to solve a problem. This information is represented in *inference structures*, which are diagrams showing how various **inferences** and **knowledge roles** link together to perform problem solving. There may be one or more inference structures in a single application.

Both KADS-I and CommonKADS provide a library of generic inference structures, indexed by the type of task which is being performed (classification, diagnosis, configuration, etc.). These generic inference structures can be used as a starting point for developing an inference structure for a particular application.

An example of an inference structure can be seen in Figure 5.

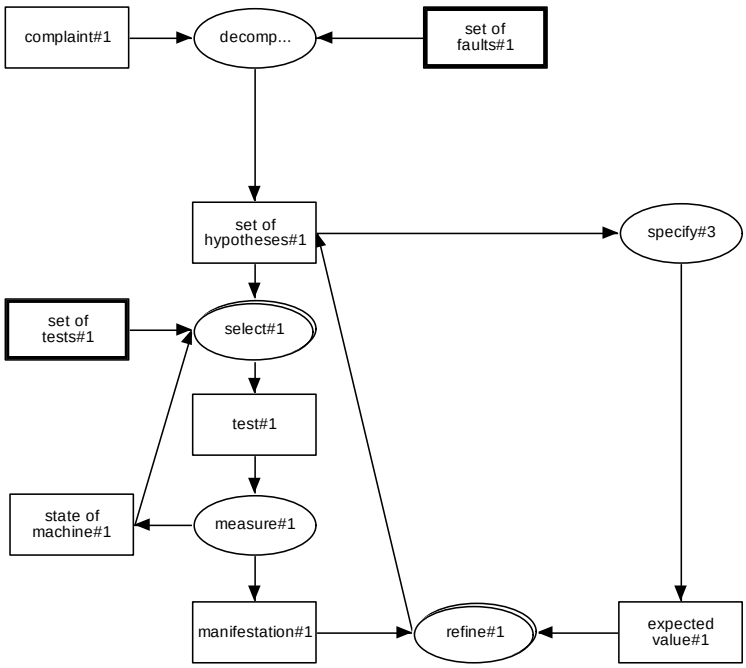

Figure 5: Inference structure: Machine Fault Diagnosis

In this inference structure, machine fault diagnosis is seen as a process of:

1. decomposing a set of possible faults, on the basis of some symptoms, to produce a set of hypotheses about the current fault;
2. selecting a suitable test to narrow down the set of hypotheses;

3. deciding what the expected result of this test would be if any of the hypotheses were correct;

4. performing the test and observing or measuring the result;

5. updating the set of hypotheses, by removing any which do not conform with the result of the test.

The last four steps are performed iteratively until only one hypothesis remains, or until no further tests are available. This inference structure therefore represents the "Sherlock Holmes" approach to diagnosis: by eliminating the impossible, whatever remains must be the truth.

It can be seen that knowledge roles are represented as boxes, and inference steps as ovals; arrows link knowledge roles with inference steps. Variations on the basic graphics include inference steps with a "double oval" (indicating that this inference step is expanded into more detail in a lower level inference structure) and bold outlines on knowledge roles (which indicate that a knowledge role is *static*; that is, the knowledge it represents is not changed in any way during problem solving.

2.5.3 Expertise Model: Task Level

The task level of the model of expertise is normally represented as a graphical hierarchy. However, textual representation is also possible, and can be more informative; see [Kingston, 1993b] for an example.

Task knowledge differs slightly between KADS-I and CommonKADS, because KADS-I has an explicit *strategy* level of knowledge, whereas CommonKADS considers strategic knowledge (now known as *problem solving knowledge*) to be orthogonal to the domain-inference-task decomposition of the Expertise Model. In both cases, however, the task knowledge which needs to be modelled consists of various tasks which have to be performed and goals which have to be achieved, and breakdowns of these tasks into subtasks. In addition, CommonKADS has an explicit category for tasks which pass information to or from the problem solving process (known as *transfer tasks*); a selection of *problem solving methods* (pre-defined generic task structures); and a detailed specification of how tasks should be defined.

An example of a task structure can be seen in figure 6.

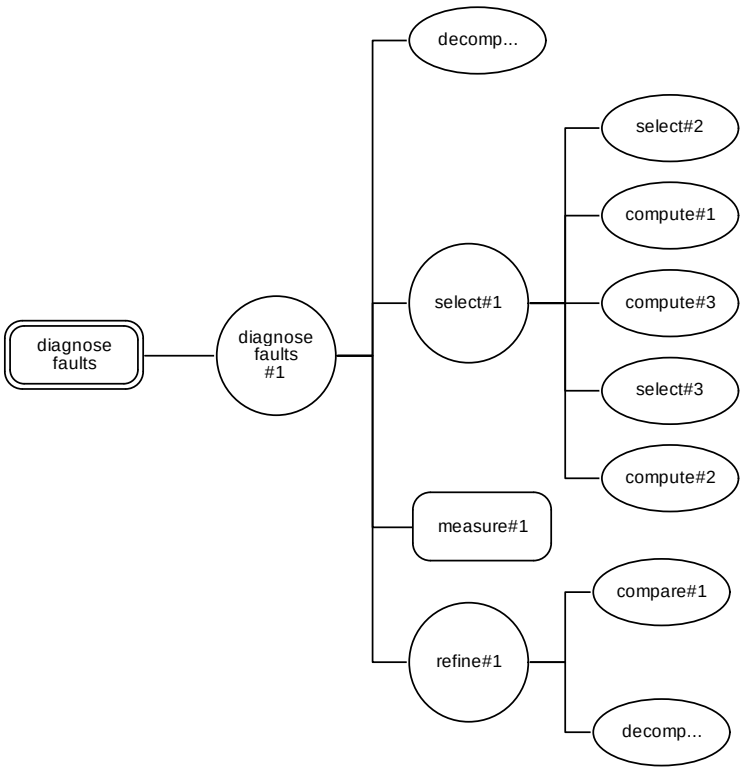

Figure 6: Task structure: Machine Fault Diagnosis

This diagram represents a decomposition hierarchy, showing the tasks which must be performed in order to diagnose faults in a machine.

CommonKADS also provides definitions for a library of *problem solving methods* i.e. predefined task structures for certain tasks. However, this "library" currently contains very few problem solving methods (cf. [Kingston, 1993a]).

2.6 Design Model

The design process in CommonKADS is broken down into three stages:

- **Application design**: decomposing the Expertise Model into a number of components. Decomposition can be functional, object-oriented, or specific to a particular AI paradigm (e.g. blackboard systems).
- **Architectural design**: deciding on the AI techniques and representations which would be most appropriate for implementing the different decomposed items.
- **Platform design**: decide how to implement these AI techniques and representations in the chosen programming tool on the chosen hardware.

While it is possible to represent the Application Design (at least) in diagrams, CommonKADS does not recommend any graphical format for the Design Model; instead, it is expected that the different stages of this model will be represented using text. However, during the course of the evaluation, a graphical representation for the Design Model was used (see figure 7); the four columns of nodes represent:

- tasks (from the Task Structure of the Expertise Model);
- functional units (the Application Design);
- knowledge representation techniques (the Architectural Design);
- particular knowledge representations and inference techniques (the Platform Design).

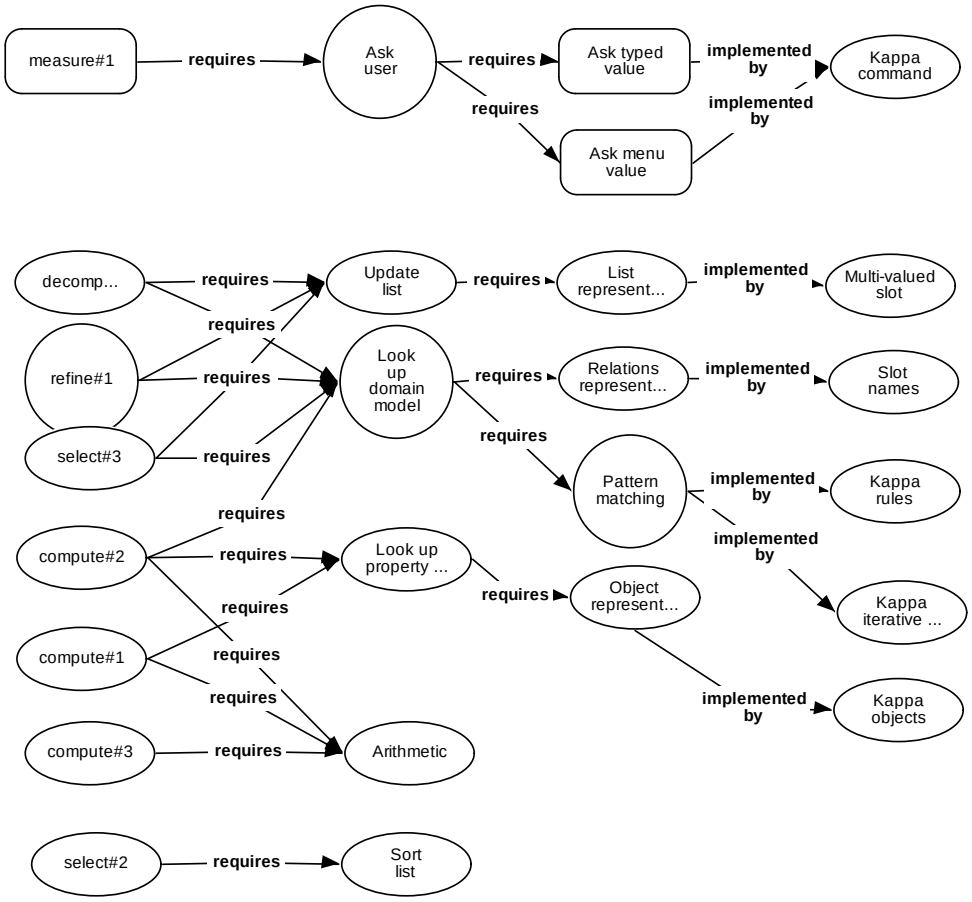

Figure 7: Design model: Diagnosing faults in machinery

2.7 CommonKADS Model Set: Summary

It can be seen that most of the models in the CommonKADS Model Set can be represented using node and arc diagrams, and/or text. Such diagrams could be drawn relatively easily with any drawing package. However, the need for a specialised workbench arises because of the need to attach information to each node, depending on its type, and the need to maintain links between nodes in different models, where one model is produced by transforming another model.

An ideal workbench should provide support for developing models (including textual annotation of knowledge objects), maintaining models, and keeping track of links between models. The level of support provided might be simply the ability to create and connect models easily, or it might include some of the (admittedly limited) guidance which is available in CommonKADS on how to go about constructing a set of CommonKADS models.

3 Overview of workbenches

This section provides an overview of the workbenches which were evaluated on the ESSI/CATALYST project:

- Version 1.1 of ILOG's KADS Tool was evaluated on Sun SPARCstations under SunOS 4.1.3. KADS Tool is implemented in Le Lisp, which is also marketed by ILOG.
- Version 1.2 of Open KADS Tool was evaluated on Sun SPARCstations under SunOS 4.1.3. Open KADS Tool is also implemented in Le Lisp.

This section also provides overviews of the CommonKADS Workbench and the VITAL workbench.

- A beta release of version 2.1 of the CommonKADS Workbench has been evaluated, running on Sun SPARCstations under SunOS 4.1.3. The CommonKADS Workbench is implemented in SWI-Prolog version 1.8.10 and PCE Release 4.7.2 for X11R5. A PC version is also expected to become commercially available in late 1995.

- The VITAL workbench has not been evaluated directly; the comments in this paper are based on publications about VITAL (particularly [Domingue *et al*, 1993]), on occasional public demonstrations, and on further information provided by one of the developers of the workbench.

3.1 Overview of KADS Tool

KADS Tool offers a number of different facilities to support the development of CommonKADS models of a knowledge based process. It achieves this through the use of diagrams, text and hypertext-style linking to represent knowledge and its inter-relationships. KADS Tool uses Le Lisp's proprietary graphics capabilities to provide its diagramming facilities. KADS Tool is designed to use a repository of knowledge objects, which may appear in several diagrams (or hypertext documents); this repository supports ease of maintenance, because changes made to a knowledge object in one diagram are immediately propagated to all other views on that object. The underlying editor for each object may reference other knowledge objects, which may exist in different levels of the Expertise Model.

KADS Tool is intended to support:

- knowledge acquisition from documents, which can include both text and graphics;
- representation of the CommonKADS Expertise Model – domain, inference and task layers.

There are no explicit facilities provided for supporting models recommended by CommonKADS other than the Expertise Model; however, it is possible to use the facilities provided for the Expertise Model to represent most of these models in a suitable form.

Release 2.0 of KADS Tool uses the Object Modelling Technique (OMT) notation [Rumbaugh *et al*, 1991] for domain modelling. This was introduced because CommonKADS uses OMT graphical notation for its domain modelling.

3.2 Overview of Open KADS Tool

Open KADS Tool is a workbench which offers a number of different facilities to support the development of KADS models of a knowledge based process. It achieves this through the use of diagrams and object-like structures to represent various items of knowledge,

and linking between them. It also supports the classification of acquired knowledge into certain ontological categories, by permitting the creation of separate "folders" for different categories, and by using different graphical notation for each category.

Open KADS is also intended to support:

- knowledge acquisition from documents, which may include both text and graphics;
- representation of the Expertise Model – domain, inference, task and strategy layers.

Open KADS Tool's facilities are intended to support modelling according to the recommendations of the KADS-I project (cf. [Anjewierden *et al*, 1990]). Open KADS Tool therefore does not support certain features of CommonKADS which have been introduced in the KADS-II project. It is therefore unsurprising that there are no explicit facilities provided for supporting any CommonKADS models apart from the Expertise Model.

Like KADS Tool, Open KADS Tool uses a repository of "knowledge objects" as the basis for its dictionaries and viewers. This means that, if a knowledge object is altered in one dictionary or viewer, the changes made are immediately propagated to all other views on that object.

3.3 Overview of the CommonKADS Workbench

The CommonKADS Workbench as the name suggests, is intended to support the CommonKADS methodology in full. This workbench was produced as part of the KADS-II project, and is expected to become commercially available (from Integral Solutions Ltd) during 1995.

The CommonKADS Workbench offers a number of different facilities to support a knowledge engineer in developing all the six models recommended by CommonKADS. It also provides some support for project management of a CommonKADS project, including support for life-cycle management, quality analysis, and document generation. This is achieved by allowing a user to request the project manager's interface or the knowledge engineer's interface. The project manager is able to determine the resources available to and tasks required from each knowledge engineer; these alterations are reflected in the interface provided when a particular knowledge engineer logs in.

The CommonKADS Workbench is also the only workbench which fully supports generation of CommonKADS' Conceptual Modelling Language (CML). The workbench includes

a utility for transforming CML into FML, the formal modelling language which was developed as a part of CommonKADS [van Harmelen & Balder, 1992].

The CommonKADS Workbench uses an incremental loading approach (facilities are only loaded into the workbench when they are required), which occupies less disk space than a single binary image, but causes some delays (typically 5-30 seconds, running on a Sun SPARCstation 1) when each facility is used for the first time.

3.4 Overview of VITAL

The VITAL workbench supports the VITAL methodology (see e.g. [Shadbolt *et al*, 1993]) which is broadly similar to CommonKADS. The VITAL workbench is an integrated software environment that aims to support the various activities, models and languages which are defined in the VITAL methodology. The workbench mirrors the decomposition of the VITAL methodology, as it is functionally organised into four *assistants*. Each assistant comprises a set of tools which can be used for producing the relevant process product. These assistants are:

- Requirements Specification Assistant
- Design Assistant
- Analysis Assistant
- Implementation Assistant

The VITAL workbench contains a meta object management system, which acts as a repository of knowledge. A simplified version of this management system is used as the intermediate repository between the several knowledge acquisition tools encoded in the VITAL workbench and its knowledge representation facilities. This repository is also interfaced with VITAL's own "conceptual modelling language" which is automatically updated to reflect the contents of the various graphical models.

The VITAL workbench is the only workbench of those reviewed in this paper which provides significant support for knowledge engineers who are implementing a knowledge base in a target language. The VITAL workbench has also paid a lot of attention to the visualisation of knowledge bases and software; as a result, it provides graphical displays which are capable of interacting with an implemented knowledge base. The VITAL workbench also provides some support for project management.

The VITAL workbench will probably not be made commercially available, although a spin-off from the workbench – a tool for constructing Generalised Directive Models (implemented in C) – is expected to be made available commercially. The Open University will also produce a PC based version of the workbench for a new Knowledge Analysis and Design course. The PC version will contain the design tools with an operational language. It will run on a 4MB 386 PC, and will be implemented in Allegro PC Lisp.

4 Evaluation of workbenches

The evaluation was carried out in the spring and summer of 1994. It was discovered that almost all the facilities available in KADS Tool and Open KADS Tool are intended to support the development of the Expertise Model. This appears to be a legacy of the KADS-I method, in which the Expertise Model was the only model which was defined in detail. Indeed, Open KADS Tool is apparently intended to support the older KADS-I approach to expertise modelling, whereas KADS Tool supports the CommonKADS approach to expertise modelling. The evaluation therefore concentrated its attentions on the quality of the features available to support expertise modelling.

The support for each of the "levels" of the Expertise Model will therefore be examined in detail. The evaluation also investigates the facilities provided by the tools for knowledge acquisition, because knowledge acquisition is essential if the CommonKADS models are to be populated, even though it is not a part of the CommonKADS methodology. Some investigations were also made to determine if the facilities provided could be used to represent other CommonKADS models.

4.1 Support for expertise modelling: domain knowledge

4.1.1 Domain ontology

All of the workbenches evaluated in this study provide or allow separate dictionaries (alphabetically ordered repositories) for different ontological types in the domain knowledge. The set of all these dictionaries forms the CommonKADS domain ontology. There are some differences in functionality between the tools, which are outlined below.

In KADS Tool domain terms can be transferred directly from acquired knowledge into the dictionaries. In Open KADS Tool however, there is an intermediate step: the creation of a glossary/lexicon of terms which have been extracted from acquired knowledge. Terms

in the lexicon can then be classified as concepts, relations, or can be left unclassified for use at other levels of modelling (e.g. they may later be classified as tasks).

The dictionaries in Open KADS Tool are actually named 'folders'. These folders can have sub-folders, which allows the division of concepts and relations into a number of sub-categories, distinguished according to any other criterion chosen by the knowledge engineer. This is a useful feature because it allows the segregation of concepts of particular types (e.g. faults, symptoms, and tests). Segregation into types is useful for building domain models (which often display all concepts of one particular type) and at the model ontology level (which might use these types as a basis for abstracting the domain level). Open KADS Tool also allows a knowledge engineer to distinguish between concepts and instances of concepts; however, it does not support a dictionary for properties. Instead, properties must be defined as attributes of concepts.

All the workbenches provide form-based editors for each ontological type. In KADS Tool and Open KADS Tool certain terms in these editors are hyperlinked to other items in the underlying repository; it is therefore possible to navigate between items in the repository and in the acquired knowledge with ease. The CommonKADS Workbench does not have quite such smooth linking, but it does provide full support for generating CommonKADS' Conceptual Modelling Language (CML), which represents CommonKADS models in textual form, from the domain level. Its form-based editors are therefore geared towards acquiring and representing the information required for CML.

Domain knowledge in the VITAL methodology consists of concepts, instances, relations and relation instances, along with hierarchies of concepts and relations.

4.1.2 Domain models

All the workbenches are capable of displaying domain models, both hierarchical models (taxonomies of concepts) or non-hierarchical "semantic networks". Given that most applications contain several domain models using a number of different relationships, the ability to create and display domain models graphically is an important feature for a KADS support workbench. KADS Tool permits domain knowledge relationships to be created and displayed graphically; Open KADS Tool only permits graphical display of relationships which have previously been defined using text editors; the CommonKADS Workbench and the VITAL workbench allow graphical display and creation of domain hierarchies, but have no graphical facilities for displaying non-hierarchical relationships.

KADS Tool is able to display tertiary or higher relations, i.e. relations which link three or more items. This is achieved by defining the appropriate number of arguments in the relation's form-based editor.

4.1.3 Model ontology and model schema

Apart from the CommonKADS Workbench which provides a form-based editor for a model ontology, none of the workbenches provide any support for representing a model ontology or model schemata. However, it is possible to use the existing features of a workbench (particularly semantic networks) to represent models which are not explicitly supported by the workbench (see section 4.5). This was the approach taken in most cases to represent the model ontology and model schemata.

4.2 Support for expertise modelling: inference knowledge

All the KADS workbenches (i.e KADS Tool, Open KADS Tool and the CommonKADS Workbench) support the development of inference structure diagrams which can be edited or added to dynamically. Inference structures can be developed from scratch, or can be drawn from the library of generic inference structures which was suggested for the KADS-I project. Specialised editors for inference steps and knowledge roles are provided, which allow inference steps to be hyperlinked to lower-level inference steps and inference structures, and knowledge roles to be hyperlinked to domain knowledge objects. The graphical inference structure editors apply certain restrictions (e.g. inference steps may only be linked to knowledge roles, not to other inference steps) which are requirements for correct inference structure modelling. Inference steps can be linked to lower-level inference structures, which provide more detailed information about the operation of a particular inference step.

In addition to the functionality described above, KADS Tool and the CommonKADS Workbench support the definition of the type of an inference (KADS Tool provides a simple typology of inference steps and knowledge roles). However, the CommonKADS Workbench offers fewer facilities for graphical editing than the other workbenches; links between inference steps and knowledge roles have to be specified in text.

Both KADS Tool and Open KADS Tool allow users to access KADS' library of inference structures; the CommonKADS Workbench, perhaps surprisingly, only stores a small part of this library. Access is reasonably simple: in KADS Tool, for example, library models

can be selected by a single mouse click on the appropriate model name, and individual models can be directly copied out of the library using the EXPORT facility. Open KADS Tool also provides a "meta-model" which provides a suggested approach to model development.

None of the workbenches support any of the CommonKADS adaptations to inference structures ([Valente & Löckenhoff, 1994]); in addition, support for CommonKADS' graphical standards is not quite complete in KADS Tool, and very restricted in Open KADS Tool. These weaknesses, though undesirable, are understandable given that CommonKADS' standards have only been defined relatively recently.

All the KADS workbenches support smooth linking between inference structures and the task knowledge, and between knowledge roles and the domain knowledge. KADS Tool and the CommonKADS Workbench also support linking between tasks and individual inference steps.

The VITAL workbench provides facilities for graphical development of Generalised Directive Models, which are similar to inference structures; each of these models can be thought of as a component of an inference structure. These components can be used to drive knowledge acquisition; there are re-write rules associated with each link in a Generalised Directive Model, which suggest the most appropriate technique for acquiring knowledge for a relationship of this type.

4.3 Support for expertise modelling: task knowledge

Task knowledge in the CommonKADS Expertise model is represented as a combination of *goals* and *methods*. An individual task has a goal (represented in the *task definition*) and one or more methods which can fulfil that goal (represented in the *task body*). Methods in turn can be defined as *composite* methods (made up of other methods and/or goals), *primitive* methods, or *transfer* tasks. Each of these subtypes has its own graphical representation within the KADS workbenches (except that Open KADS Tool does not support any representation for transfer tasks) and its own form-based editor. Both goals and methods can be linked to inference structures, or to individual inference steps.

KADS Tool and the CommonKADS Workbench permit the display of hierarchies of goals and methods, which show the decomposition of one or more goals or methods into subgoals or methods. Open KADS Tool chooses to divide the task knowledge between the "strategic knowledge" (see section 2.5.3), which defines goals, composite methods,

and task hierarchies, and the "task knowledge" which represents primitive methods and their links to the inference knowledge.

The VITAL workbench provides facilities for representation of tasks, which may have component tasks and goals. It also provides a separate "problem solving architecture" for specifying control of tasks, and flow of data between tasks. These two facilities fulfil a similar role to the role which the task knowledge plays in a CommonKADS Expertise model; they can also be used for specifying the design of the system (i.e. in place of the CommonKADS Design Model).

4.4 Support for knowledge acquisition

Given that interviewing is the most popular technique for knowledge elicitation, a tool for viewing transcripts would be the minimum requirement for knowledge elicitation. Support for smooth transfer of relevant terms from a transcript to the expertise model would be appreciated, as would the existence of links between the transcript and the expertise model to facilitate indexing, and the ability to update or annotate the transcript dynamically. An ideal tool might be able to identify some relevant items automatically using simple natural language analysis (e.g. identifying properties and values by parsing for adjectives associated with existing concepts). Support for the other knowledge elicitation techniques, if any, should follow the same pattern.

All of these workbenches support a transcript editor which can include a textual transcript and/or graphical diagrams. Fragments of text (and diagrams, in KADS Tool and Open KADS Tool) can be marked up as "interesting" items, and hyperlinked to various ontological types (such as concepts or relations). It is possible to traverse these links in both directions, which supplies a facility for indexing knowledge. However, no tool supports any natural language analysis of transcripts.

KADS Tool allows fragments of graphical diagrams to be hyperlinked to the domain ontology. Currently, this facility is only supported for diagrams created using KADS Tool's own diagramming format; it is hoped that it will eventually be possible to apply this technique to diagrams in other formats.

The VITAL workbench stands out from the other workbenches in the area of knowledge acquisition tools. In addition to supporting transcript analysis in a similar fashion to the KADS workbenches, it provides graphical support for the laddered grid technique, the card sort technique and the repertory grid technique, along with a common underlying repository which allows the output of one technique to be used as the input for another.

4.5 Support for other CommonKADS models

The CommonKADS Workbench is the only workbench which provides specialised editors for creating the CommomKADS Organisational, Task, Agent, Communication and Design models. However, it proved possible to use the facilities provided for expertise modelling in KADS Tool and Open KADS Tool to represent most of knowledge which is contained in other models. The various perspectives of the Organisational model can be represented using concepts and semantic networks; the Task model can be represented using the task structure editor; the Agent and Communication models can be represented using semantic networks; and the various specifications performed in the Design model were represented in a single semantic network, with attached text which made use of a specified Program Definition Language to define the details of each operation. An example of a semantic network which has been used to represent a Design model is shown in Figure 7.

In the VITAL workbench, the representation of tasks and the problem solving architecture can be used as a design specification. The VITAL workbench also includes support for explict representation of design rationale. This is achieved by augmenting design models with a means for recording argumentation. More details can be found in [Stutt & Motta, 1995].

5 Capabilities of the workbenches

A good knowledge engineering workbench needs more than just expressive power; it must also be a good piece of software with good user interfaces. The evaluation which was carried out examined the technical capabilities of each workbench using a checklist which was derived from sources such as [DTI/NCC, 1987] and [CCTA, 1990]. Some of the main findings from this evaluation are summarised below.

5.1 Technical capabilities: interfaces

All the workbenches described in this paper are primarily intended for supporting a user in modelling, rather than in integrating information from different sources. This means that they require a good interface with the user, but there is not such a strong requirement for good interfaces with other software packages. The latter were therefore not evaluated.

All tools support a range of report output formats; PostScript and ASCII are usually available, LaTeX and RTF may also be available. KADS Tool has also been linked to Kennedy-Carter's "Intelligent OOA" Case Tool. Linking to other tools may be considered in the future.

The tools all use multiple windows to represent different viewpoints on the knowledge. In KADS Tool, Open KADS Tool and the CommonKADS Workbench, four main types of window are supported:

Dictionaries for viewing the objects which constitute the expertise model. There is a dictionary for every object type which the tool supports.

Form-based editors for editing the contents of all objects. These editors are "viewers" for the contents of the objects stored in the various dictionaries. Every object type has its own editor. These editors permit descriptions and annotations to be added to objects.

Graphic editors provide an additional graphical view of the expertise model. These may include graphical views of domain hierarchies, semantic networks, inference structures, inference hierarchies and task hierarchies. Many graphic editors permit interactive creation of objects and their display at different levels of detail, e.g. concepts can be displayed with or without their associated properties.

VITAL has many facilities for visualisation. It provides a framework [Domingue *et al*, 1992] and programming language [Domingue *et al*, 1994] for defining software visualisation systems. The framework, called Viz, has the following components:

- Histories: a record of key events that occur over time as the program runs, with each event belonging to a player; each event is linked to some part of the code and may cause a player to change its state.
- Views: the style in which a particular set of players, states or events is presented, such as using text, a tree, or a plotted graph; each view uses its own style and emphasises a particular dimension of the data that it is displaying.
- Mappings: the encodings used by a player to show its state changes in diagrammatic or textual form on a view using some kind of graphical language, typography, or sound; some of a player's mappings may be for the exclusive use of its navigators.

- Navigators: the tools or techniques making up the interface that allows the user to traverse a view, move between multiple views, change scale, compress or expand objects, and move forward or backward in time through the histories. A visualization can be defined using the metaphor of players, states, events, views and mappings.

Within VITAL Viz has been used to create visualizations for two forward chaining rule based systems, a commercial Prolog system, the VITAL knowledge representation framework [Gaspari & Motta, 1994], the VITAL task design interface, and a subset of a commercial Lisp system. It has also been used to create visualisations for specific applications.

General purpose & navigational windows for navigating the expertise model and performing other procedures, including validation and export. The CommonKADS expertise model is comprised of a number of objects and composite objects, which can be referenced or contained in other objects; hence the need for navigation facilities.

Figure 8 shows an example of a KADS Tool dictionary window. All windows are known as *panels* within KADS Tool.

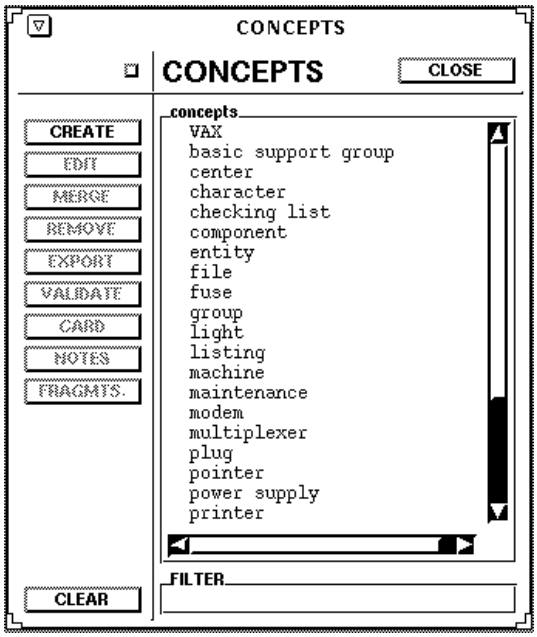

Figure 8: Example of a panel

It is relatively easy to define objects and create models; model creation is helped by the fact that every object has a single underlying representation that can be accessed

from a number of different editors. Editors can be invoked from within other editors; in fact, many editors can be invoked from *any* explicit reference to an object, including text fragments.

Some tools perform a level of syntax checking within editors which helps prevent inconsistent or incorrect models from being developed. For example, if the user attempts to create a link between two inference steps in an inference structure, none of the KADS support tools will permit this. Likewise, the user is informed if any attempts are made to modify an object definition in a way which is inconsistent with existing objects.

Further additions to the user interfaces of the tools which would be beneficial might include a zoom facility for graphical editors, and providing accelerator keys for specific operations to assist experienced users.

Project Manager's interface The CommonKADS Workbench and the VITAL workbench both provide some facilities for supporting project management. In the CommonKADS Workbench, this is achieved by providing the project manager with:

- a Lifecycle Browser;
- a Model Set Browser;
- a Project Team Browser;
- a Documentation Set Browser.

These interfaces allow a project manager to define what deliverables should be produced at each stage of a project, and who should be responsible for producing them.

The interface facilities in VITAL require considerable detail to describe, since the developers of the VITAL workbench have put a lot of effort into providing good visualisations, not only for viewing models of knowledge used at the analysis stage, but also for examining the design stage, and even for supplying information on the operation of a system to support the implementation stage. For further details, see [Domingue & Watt, 1991].

5.2 Other aspects

KADS Tool and Open KADS Tool proved to be reliable, robust and resistant to misuse – the automatic checking of inputs and the graphical method used to create relations

between objects minimise the chances of misuse. While the version of the CommonKADS Workbench which we evaluated did exhibit some bugs (nearly all recoverable errors), and some oversensitivity to misuse, this is to be expected in a beta test version of the software.

Security & Integrity None of the workbenches offer any features for restricting access (read and/or write) to an application or part of it, e.g. requiring password clearance. However, it is possible to restrict access to files within Unix, using file permissions, modes and ownership. In addition, read only libraries can be created and re-used in a number of applications. The CommonKADS Workbench allows the project manager to specify which tasks should be carried out by which knowledge engineers.

Quality control KADS Tool and Open KADS Tool provide good support for version control, and for restoring previous versions of an application. The beta version of the CommonKADS Workbench does not provide any such facilities.

Quality control is particularly important if an application is being developed by more than one knowledge engineer. Open KADS Tool provides support for multiple users by providing facilities for structuring applications into universes and folders. A universe, set of folders or an entire application can be 'published', i.e. exported and saved to a file. Any number of 'publications' can be integrated together to form a new application. Objects which have the same name and description are merged together. This makes it possible to distribute an application with a set of 'core' objects, e.g. a library, to a number of knowledge engineers, who can develop their own 'publications' separately. All extensions of the application can then be merged into the one application.

KADS Tool provides support for multiple users through module management. An application is composed of a set of *knowledge modules* and a module is composed of a set of KADS Tool objects. Module management allows the grouping of knowledge into consistent and homogeneous modules and the reuse of existing modules in different applications. Modules can be developed and maintained independently and by different knowledge engineers. Module management also provides a degree of security in multi-user projects because modules can be set to be 'read-only'.

The CommonKADS Workbench provides support for multiple users by allowing the project manager to specify which parts of a model individual users should be working on.

The VITAL workbench does not provide any facilities for multiple users, although plans exist for introducing such facilities, if the workbench is developed further.

Documentation All the tools provide a User guide, which is generally the most useful document for those trying to get to grips with a workbench. The User Guides for KADS Tool, Open KADS Tool and the CommonKADS Workbench are all well-written and generally usable. In addition, KADS Tool provides a Methodological guide and the CommonKADS Workbench provides a guide to the basic facilities of the tool, and a guide to providing a demo of the tool. Our evaluation found the Demo guide to be useful at the very beginning of using the CommonKADS Workbench, but the other documents were not particularly helpful for the first-time user.

Platforms All the workbenches described in this section are available on UNIX workstations. In addition, the CommonKADS Workbench is available on a PC under Windows, and a version of the Generalised Directive Model component of VITAL will become available for a PC.

6 Future enhancements

Open KADS Tool

Since this evaluation was performed, Bull have announced release 2.0 of Open KADS Tool. This will include the following changes:

- support for design modelling, which will encapsulate code files in Open KADS Tool, thus allowing traceability between the knowledge model and the design model;
- customisable code generation (of C++, KOOL, G2, and ROCK);
- verification of user-defined syntax within the expertise model;
- pre-defined solutions (Open KADS Tool models, C++ libraries, and graphical interfaces) for some process control and diagnostic problems.

Planned enhancements for future releases include:

- the provision of validation facilities (based on work in the VITAL project, in which Bull is a contributor); and
- vertical models, i.e. models that include some domain, inference, task knowledge and which needs to be specialised for a particular application.

KADS Tool

As mentioned above, release 2.0 of KADS Tool uses the OMT notation. No further information on future enhancements is currently available.

The CommonKADS Workbench

It is hoped that future versions of the workbench will allow automatic generation of executable code.

VITAL

No enhancements to VITAL are expected, although porting of part of the workbench to a PC platform and of the ideas from VITAL to Open KADS Tool (see section 6) are expected to take place.

7 Summary

From the above analysis, it can be seen that all the workbenches reviewed are capable of supporting CommonKADS development projects, although the VITAL workbench is much better suited to supporting the VITAL methodology than to supporting CommonKADS. However, some offer better facilities than others in certain areas:

- KADS Tool provides the best facilities of all the workbenches for graphical creation of models, and also supports features of the Expertise Model which were introduced in CommonKADS. However, it does not support the model ontology level of the domain knowledge, nor does it support any other CommonKADS models.
- Open KADS Tool supports subdivision of domain dictionaries into folders, which could be very useful on larger projects, and also has more features explicitly intended for multi-user support. However, it is based on KADS-I rather than CommonKADS, so it lacks support for a number of features which were introduced in CommonKADS.
- The CommonKADS Workbench supports all the CommonKADS models, rather than the Expertise Model alone. It also provides some support for CommonKADS project management. However, its current version is slower to start up than the

other tools, because its facilities are loaded incrementally, rather than being loaded as a binary image.

- The VITAL workbench provides good facilities for knowledge acquisition, and supports design and implementation with visualisations of the processes involved; it also provides some support for project management. However, it does not attempt to provide support for many features of CommonKADS, since it is based on a different methodology. It is also uncertain whether the VITAL workbench will ever become commercially available.

The requirements of an organisation will determine the most appropriate tool for that organisation. Factors to consider include:

- whether it is considered important to use CommonKADS rather than KADS-I;
- whether the full set of CommonKADS models will be developed rather than just an Expertise Model;
- whether support for implementation is needed as well as for analysis and design.

The appendices to this paper provides a summary of the features of KADS Tool, Open KADS Tool and the CommonKADS Workbench.

References

[Anjewierden *et al*, 1990] Anjewierden, A., Wielemaker, J. and Toussaint, C. (1990). Shelley – Computer Aided Knowledge Engineering. In *Current Trends in Knowledge Acquisition*. IOS Press.

[Breuker & van de Velde, 1994] Breuker, J. and van de Velde, W. (1994). *The CommonKADS Library: reusable components for artificial problem solving*. IOS Press, Amsterdam.

[CCTA, 1990] CCTA. (1990). *ISE Appraisal and Evaluation Library*. HMSO.

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

[deHoog *et al*, 1993b] de Hoog, R., Martil, R., Wielinga, B., Taylor, R., Bright, C. and van de Velde, W. (1993). The Common KADS model set. ESPRIT Project P5248 KADS-II/M1/DM1.1b/UvA/018/5.0, University of Amsterdam & others.

[Domingue & Watt, 1991] Domingue, J. and Watt, S. (Dec 1991). Graphical User Interface and Windowing in the VITAL Project. VITAL Series 84, Human Cognition Research Laboratory, The Open University, Walton Hall, Milton Keynes, MK7 6AA.

[Domingue *et al*, 1992] Domingue, J., Price, B. and Eisenstadt, M. (1992). Viz: a framework for describing and implementing software visualization systems. In Gilmore, D. and Winder, R., (eds.), *User-Centred Requirements for Software Engineering Environments*. Springer-Verlag.

[Domingue *et al*, 1993] Domingue, J., Motta, E. and Watt, S. (Sept 1993). The Emerging VITAL Workbench. In *Knowledge Acquisition for Knowledge-based Systems: 7th European Workshop EKAW '93*, pages 320–339, Toulouse and Caylus, France. Springer-Verlag.

[Domingue *et al*, 1994] Domingue, J., Eisenstadt, M. and Price, B. (1994). The VITAL visualization tool report. VITAL Project Report DD342, HCRL, Open University.

[DTI/NCC, 1987] DTI/NCC. (1987). *STARTS guide*, volume 1. NCC publications.

[Gaspari & Motta, 1994] Gaspari, M. and Motta, E. (1994). Symbol-level requirements for agent-level programming. In *Proceedings of the 11th European Conference on Artificial Intelligence, ECAI '94*.

[Kingston, 1993a] Kingston, J.K.C. (1993a). Pragmatic KADS 1.0. Technical Report AIAI-IR-13, AIAI, University of Edinburgh.

[Kingston, 1993b] Kingston, J.K.C. (1993b). Re-engineering IMPRESS and X-MATE using CommonKADS. In *Expert Systems 93*. British Computer Society, Cambridge University Press. Also available as AIAI-TR-130.

[Motta *et al*, 1995] Motta, E., O'Hara, K., Shadbolt, N., Stutt, A. and Zdrahal, Z. (1995). Solving VT in VITAL: A Study in Model Construction and Knowledge Reuse. *International Journal of Human Computer Studies*.

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

[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.

[Shadbolt *et al*, 1993] Shadbolt, N., Motta, E. and Rouge, A. (Nov 1993). Constructing Knowledge Based Systems. *IEEE Software*.

[Stutt & Motta, 1995] Stutt, A. and Motta, E. (1995). Recording the design decisions of a knowledge engineering community to facilitate re-use of design models. In *Proceedings of the 1995 Knowledge Acquisition Workshop*, Banff, Canada.

[Tansley & Hayball, 1993] Tansley, D.S.W. and Hayball, C.C. (1993). *Knowledge-Based Systems Analysis and Design: A KADS Developers Handbook*. Prentice Hall.

[Valente & Löckenhoff, 1994] Valente, A. and Löckenhoff, C. (1994). Assessment. In Breuker, J. and van de Velde, W., (eds.), *The CommonKADS Library*, chapter 8, pages 169–190. KADS-II Consortium, KADS-II/TM.2/VUB/TR/054/3.0. This document is also available as a book from IOS Press.

[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'.

[Waern *et al*, 1994] Waern, A., Höök, K., Gustavsson, R. and Holm, P. (1994). The Common KADS Communication Model. ESPRIT Project P5248 KADS-II KADS-II/M3/TR/SICS, Swedish Institute of Computer Science.

[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)*.

A Summary: KADS Tool

A.1 Model Of Expertise

KADS Tool supports sufficient notations (both textual and graphic) for full representation of the CommonKADS Expertise Model. Model development is supported very well. The components of the expertise model will be evaluated individually in the following sections.

A.1.1 Domain Knowledge

Coverage and Compliance with CommonKADS

Concepts	Dictionary and form-based editor.
Properties	Dictionary and form-based editor. Properties can be assigned to concepts and relations.
Attributes	No explicit support. However, attributes can be represented as a concept with a single property, called *value*.
Relations	Dictionary and form-based editor. Two pre-defined relation types are provided: "sub type" and "composed of". User defined relationships can also be used and properties can be assigned.
Expressions	A dictionary and form-based editor are provided. The editor provides support for constructing the definition but not for checking its syntax.
Domain Models	Semantic net and domain hierarchy editors are provided.

Support for Visualisation and Development

Visualisation	Domain hierarchies can be displayed graphically using the domain hierarchy editor and relations between domain objects can be displayed using semantic networks. The recommended OMT (Object Modelling Technique) graphic formalism is used. The properties of concepts and their values can be displayed in the semantic networks and domain hierarchy editors.
Development support	Domain models can be developed using both the form-based *and* the graphical editors. All editor types are well integrated and are intuitive to use.

A.1.2 Inference Knowledge

Coverage and Compliance with CommonKADS

Knowledge roles	Dictionary and form-based editor. Dynamic and static roles are distinguished, though role sets are not distinguished from single-item knowledge roles. Knowledge roles can be mapped to objects in the domain layer, e.g. concepts, semantic nets, etc, but not to the task layer.
Inference steps	Dictionary and form-based editor.
Inference structures	Dictionary and specialised graphic editor. Inference structures can be decomposed hierarchically; with an individual inference step being expanded into an entire inference structure. Transfer tasks cannot be included as components in inference structures.

Support for Visualisation and Development

Visualisation	Inference structures can be displayed using a specialised graphical editor. Inference and role annotations can be displayed graphically in the inference structure. The recommended CommonKADS graphical notation is adhered to, apart from composite inferences and static roles.
Development tools	Libraries of knowledge role types, inference steps and inference structures are provided. All editors are well integrated, are intuitive and easy to use.

A.1.3 Task Knowledge

Coverage and Compliance with CommonKADS

Task Goals	Dictionary and form-based editor.
Task Methods	Dictionary and form-based editor.
Task hierarchies	Specialised graphic editor for displaying and creating task hierarchies.

Support for Visualisation and Development

Visualisation	Task hierarchies can be displayed graphically. The tree structure can be displayed vertically, horizontally and in an abbreviated form. Parts of the hierarchy can be "hidden" if required. The distinction between AND & OR hierarchies is implicit in the tree structure.
Development tools	Task hierarchies can be created graphically. All editors are well integrated, are intuitive and easy to use.

A.1.4 Problem Solving Knowledge

Some support for problem-solving knowledge is provided at the task level. This support allows the definition of strategic knowledge which specifies how to select between alternative task methods. This knowledge can be described in the form-based editor for the task goal.

A.1.5 Inter-category links

Coverage and Compliance with CommonKADS

Inter-category links are available between:

- Domain knowledge and inference knowledge: knowledge roles can be "mapped" onto any domain object.
- Task knowledge and inference knowledge: inference structures and inference steps can be linked to goals/methods.

Support for Visualisation and Development

Creating links between the various categories of knowledge is straightforward using the form-based editors. The inference structure editor can display the domain objects which are mapped onto the knowledge roles. It does not display the goals or tasks which activate the inference steps. Likewise, the mappings from the task category to the inference category are not displayed graphically.

It is not possible to display objects from all categories in a single window, nor to create or visualise the inter-category links graphically.

A.2 Support for knowledge acquisition

A.2.1 Transcript Analysis

Transcript editing	A text *and* diagram editor are provided for creating transcript documents. The graphics editor has a set of basic drawing tools and can import GIF, PBM, PMM, PGM, and DXF graphics types.
Transcript linking	Good. One or more words or any part of a diagram can be identified as a "fragment", and linked to any new or existing dictionary item. Different link types are distinguished using a combination of colours and font types. However, it is not possible to distinguish between the link types within a knowledge layer, e.g. between concepts and properties (although this has been amended in the latest release of Open KADS Tool). It would be useful to have an intermediate store, e.g. a glossary, for interesting fragments which are pending classification.
Ease of Use	Good. Transcripts are easy to create and annotate (i.e. link to dictionary items). Navigating links between fragments and dictionary items is relatively straightforward.

A.2.2 Other KA techniques

No other techniques, e.g. card sorting, repertory grid analysis, are supported.

A.3 Support for other CommonKADS models

Organisational	The various perspectives of this model can be represented using a combination of semantic networks and form-based editors. The semantic networks may contain concepts, goals, tasks and relations.
Task	This model can be represented using a combination of the facilities for modelling task knowledge in Expertise Model and semantic networks (for modelling precedence links).
Agent	This model can be represented using free-form text in the "description" of a task. It could also be represented in a semantic network as concepts related to tasks; the less obtrusive option of cross-referencing objects is not available between concepts and tasks.
Communication	This model can be represented as textual annotation of the Task Model, or as an extension to a semantic network representing an Agent model.
Design	This model can be represented in a semantic network, using tasks and/or concepts with specified relations.

A.4 Maintainability

Object storage	Objects are stored in a central repository; modifications to an object are reflected immediately and automatically in all relevant editors.
Change propagation	Changes made to the names of objects are reflected in the definition of all other objects where it is cross-referenced.
Syntax checking	Checking for naming conflicts and some syntactic errors is performed on all object editors (except on the definitions of an expression and a method body) during model building.
Verification	Explicit validation procedures for redundancy, completeness and consistency can be invoked for all objects and models.

A.5 Other features

Reliability	There were some minor bugs in the evaluation version. However, these should be sorted out in version 2.0.
Resistance to misuse	The method of interaction helps to make KADS Tool resistant to misuse.
Encryption	No facilities for encrypting information in parts of the models, e.g. for reasons of security/confidentiality.
Integrity	Modules can be made "read only", although this does not require any security clearance and can be modified by any user.
Multi-users	An application is composed of a set of *knowledge modules*, which can be developed and maintained independently and by different knowledge engineers. A level of security is provided for modules. Facilities for navigating and viewing the modular structure of an application are adequate, but can be cumbersome to use.

A.6 Quality Control

Version control	Automatic creation of new versions, if requested. Facilities for restoring previous versions of an application or module.
Change control	Facilities for restoring any previous version of an application and also for warning of implications of importing an object or module. However, there is no global undo facility.
Modularity	Modules can be developed separately and independently and can be shared between applications. Facilities exist for exporting and merging objects.
Incremental development	Module management, facilities for export and merging of modules and applications and the automatic updating of cross-references between objects provide support for incremental development.
Documentation	No facilities are provided for automatic documentation of the above.

A.7 Documentation

User guide	Clear and easy to understand. Reasonable indexing, although one or two terms which could usefully be in the index are not there (such as *inheritance*).
Methodology guide	Not very comprehensive. Does not have an index.
Tutorial guide	Available only through training course.
On-line help	None exists.

A.8 Costs

These costs are correct as of July 1994, using the exchange rate in force at that date.

KADS Tool (single user)	£8,600
12 months maintenance	£1,290
KADS Tool training (1 day course)	£350 (per person)

1. all prices above are exclusive of VAT;
2. price of KADS Tool includes a licence for LeLisp;
3. quantity discounts are available for multiple purchases of KADS Tool:
   - 10 % for quantities between 2 to 4;
   - 20 % for quantities 5 to 9.
4. licence is a token based floating one and is not restricted to a specific machine.

B Summary: Open KADS Tool

B.1 Model Of Expertise

Open KADS Tool supports sufficient notations (both textual and graphic) for adequate representation of the CommonKADS Expertise Model. Model development is supported well. The support provided for the three levels of the expertise model will be described in the following sections.

B.1.1 Domain Knowledge

Coverage and Compliance with CommonKADS

Concepts	A form-based editor is provided and folders can be used to categorise concepts. Instances of concepts are supported.
Properties	Not explicitly supported. KADS-I "attributes" provide most of the required functionality; however, it is not possible to assign "attributes" to relations and there is no specialised editor for "attributes".
Attributes	Not explicitly supported (*KADS-I "attributes" are not the same as those defined for CommonKADS*). However, the required functionality could be supported using concepts and KADS-I "attributes".
Relations	A form-based editor is provided and folders can be used to categorise relations. One pre-defined relation type is provided: "specialization of" (semantically equivalent to IS A). This relation only allows inheritance of "attributes" between concepts, and multiple inheritance is not supported. User defined relationships can also be created, but it is not possible to specify "axioms" in order to define a relation's semantics (e.g. inheritance).
Expressions	Not supported. They could be represented using free-form text in the description field for attributes or relations.

Support for Visualisation and Development

Visualisation	Taxonomies of concepts and relations can be displayed graphically. The *Graph* facility displays links between: concepts/relations $\leftrightarrow$ attributes/fields $\leftrightarrow$ value types. Parent and children concepts/relations can also be displayed on the graph. This effectively displays *all* defined links from any domain item to other domain items. The Object Modelling Technique graphic formalism is *not* complied with in any of the domain representations.
Development tools	Domain models can be developed using the form-based editors; it is not possible to create models graphically. For example, although it is possible to drag and drop concepts between two different Hierarchy editors, it is not possible to create a sub-type link between two concepts using mouse and menu. All form-based editors are well integrated and are intuitive to use.

B.1.2 Inference Knowledge

Coverage and Compliance with CommonKADS

Knowledge roles	A dictionary and form-based editor are provided. However, dynamic and static roles are not distinguished and there is no support for identifying knowledge roles which represent sets of objects, or hierarchies of knowledge roles.
Inference steps	Referred to as "Inference Operators". A dictionary and form-based editor are provided.
Inference structures	A dictionary and specialised graphical editor are provided. The graphic editor allows knowledge roles, inference steps and links between them to be created using mouse and menu. Inference structures can be decomposed hierarchically. Transfer tasks cannot be included as components of inference structures.

Support for Visualisation and Development

Visualisation	Inference structures can be displayed using a specialised graphical editor. There are no facilities to display operator and role annotations graphically in the inference structure. The graphical notation is the same as that defined for CommonKADS, apart from composite inference structures and static knowledge roles.
Development tools	Libraries of knowledge role types, inference steps and inference structures are provided. Knowledge roles *and* inference steps can be mapped both to objects in the domain layer; inference structures can be mapped to the task knowledge. All editors are well integrated and are intuitive and easy to use. However, the development support for hierarchical inference structures is slightly awkward to use, because it restricts top-down development of hierarchical inference structures.

B.1.3 Task Knowledge

Coverage and Compliance with CommonKADS

Task goal	Task goals are represented using "Goals", which can be compound or atomic. A dictionary and form-based editor are provided. Note: Goals are deemed to be part of the KADS-I "Strategy" layer - see section 2.5.3.
Task body	Task bodies are represented using "Tasks", which have a dictionary and form-based editor.
Task hierarchies	A specialised graphics editor is provided for displaying and creating goal hierarchies. Links can be created between goals and goals and between goals and tasks.

Support for Visualisation and Development

Visualisation	Goal hierarchies can be displayed graphically; AND/OR hierarchies are explicitly distinguished. Tasks cannot be displayed in a graphical editor; the tasks required to satisfy a goal must be specified in the description field of a goal.
Development tools	The form-based editors are well integrated and easy to use. However, it is not possible to create goal hierarchies graphically.

B.1.4 Problem Solving Knowledge

Some support for problem-solving knowledge is provided at the strategy level. The strategic knowledge can be specified in the task goal (Goal) form-based editor. The strategic knowledge should specify how to select between alternative (problem-solving) task bodies.

B.1.5 Inter-category links

Coverage and Compliance with CommonKADS

Knowledge roles and inference steps can be "mapped" onto any domain object. Tasks can be linked to inference structures; however, individual inference steps or dynamic roles cannot be linked to the task knowledge.

Support for Visualisation and Development

Creating links between the various categories of knowledge is straightforward using the form-based editors. The inference structure editor cannot graphically display information about links to the domain objects or links to tasks; however, a powerful facility is provided for displaying objects from all categories in a single window and for graphically visualising the inter-category links.

B.2 Support for Knowledge acquisition

B.2.1 Transcript Analysis

Transcript editing	No facilities are provided for creating transcripts. However, transcripts can be created using standard text editors. Graphics in transcripts are also supported.
Transcript linking	One or more words or an entire diagram can be identified as a "fragment", and linked to any existing term in the glossary (stores interesting fragments pending classification) or item in any dictionary. Creating links is straightforward and different link types can be distinguished using a combination of colours and font types; link types within a knowledge category, e.g. concepts, relations, can be distinguished. The display of link types can be configured by the user.
Ease of Use	Good. Transcripts are easy to annotate (i.e. link to dictionary items or glossary terms). Navigating links between fragments and dictionary items is relatively straightforward.

B.2.2 Other KA techniques

No other techniques, e.g. card sorting, repertory grid analysis, are supported.

B.3 Support for other CommonKADS models

Organisational	This model can be explicitly represented using the domain modelling constructs. The graph facilities are useful for displaying this information graphically.
Task	This model can be explicitly represented using a combination of the facilities for goals and tasks.
Agent	This model can be represented using free-form text in the "description" of a task. It could also be represented in a semantic network as concepts related to tasks. Cross-referencing is not available between concepts and tasks.
Communication	This model can be represented as textual annotation of the Task Model, or as an extension to a semantic network representing an Agent model.
Design	This model can be represented with a combination of goals and tasks. Version 2 of Open KADS Tool should provide additional support; see section 6.

B.4 Maintainability

Object storage	Objects are stored in a central repository; modifications to an object are reflected immediately, but not always automatically in all relevant editors; editors that are open may have to be manually updated.
Change propagation	Changes made to the names of objects are reflected in the definition of all other objects where it is cross-referenced. However, this does not include information represented in text in the "description" field of objects.
Syntax checking	Checking for naming conflicts and some syntactic errors is performed on all object editors. No syntax checking is done on any information represented in the "description" field of objects.
Verification	There are no explicit validation procedures to check for redundancy, completeness of an object or consistency between objects. However, there is a facility for checking if inter-category links exist for all objects; this can also be checked visually.

B.5 Other features

Reliability	Open KADS Tool proved to be very reliable and did not crash once during the evaluation period.
Resistance to misuse	Open KADS Tool proved to be very robust and resistant to misuse; the use of form-based editors rather than typed commands minimises the chances of misuse.
Encryption	No facilities for encrypting information in parts of the models, e.g. for reasons of security/confidentiality.
Integrity	There are no features for restricting access (read and/or write) to an application or part of it, e.g. requiring password clearance. However, read-only libraries can be created and re-used in a number of applications.
Multi-users	An application is composed of a set of universes, which in turn can be structured in to a hierarchy of folders. A universe, set of folders or an entire application can be 'published', i.e. exported and saved to a file and any number of 'publications' can be integrated together to form a new application. Facilities for navigating and viewing the modular structure of an application are good.

B.6 Quality Control

Version control	No explicit support. New versions can be created explicitly using the "Save As ..." feature, which then prompts for a new name. There are no facilities for automatically restoring previous versions of an application or publication.
Change control	There are no facilities for managing change, apart from version control described above.
Modularity	Applications consist of folders which can be organised hierarchically. Folder management provides a flexible way of organising an application, which is particularly important for multi-user projects.
Incremental development	The object-based approach to modelling, automatic updating of cross-references and folder management serve to facilitate incremental development.
Documentation	No facilities are provided for automatic documentation of the above.

B.7 Documentation

User guide	Clear and easy to understand. Reasonable indexing.
On-line help	Fairly good on-line help is available.

B.8 Costs

Open KADS Tool V1.2	£5 900 (50 000 FF)
Open KADS Tool V2.0 (single user)	£11 800 (100 000 FF)

12 months maintenance

Open KADS Tool V1.2	£710 (6 000 FF)
Open KADS Tool V2.0	£1420 (12 000 FF)
Open KADS Tool training (1 day course)	£710 (6 000 FF) (per person)

1. price of Open KADS Tool includes a licence for LeLisp;
2. upgrade from V1.2 to V2.0 costs £3 550 (30 000 FF);
3. licence is a fixed host one.

C Summary: the CommonKADS Workbench

C.1 Model Of Expertise

The CommonKADS Workbench supports notations for full representation of the CommonKADS Expertise Model. Model development is fairly well supported. The components of the expertise model will be evaluated individually in the following sections.

C.1.1 Domain Knowledge

Coverage and Compliance with CommonKADS

Concepts	Dictionary and form-based editor.
Properties	Dictionary and form-based editor. Properties can be assigned to concepts and relations.
Attributes	Dictionary and form-based editor.
Relations	Dictionary and form-based editor. Two pre-defined relation types are provided: "sub type" and "composed of". User defined relationships can also be used and properties can be assigned.
Expressions	A dictionary and form-based editor are provided. The editor provides a little support for constructing the definition, but not for checking its syntax.
Domain Models	A single domain hierarchy for each ontological type exists by default. Non-hierarchical domain models can only be created and viewed using form-based editors.
Model Ontology	A form-based editor is provided for representing the model ontology, which can be produced by abstracting from the domain ontology.

Support for Visualisation and Development

Visualisation	The domain hierarchies can be displayed graphically; other domain knowledge can be displayed in form-based editors. Those graphical facilities which are provided use the recommended OMT (Object Modelling Technique) graphic formalism.is used. Properties of concepts are automatically displayed displayed in the graphical display of the concept hierarchy.
Development support	The form-based editors support some mouse-based interaction, including the ability to select from a menu which is drawn from a domain level dictionary.

C.1.2 Inference Knowledge

Coverage and Compliance with CommonKADS

Knowledge roles	Dictionary and form-based editor. Dynamic and static roles are distinguished. Role sets are supported. Roles can be mapped to objects in the domain layer, e.g. concepts, semantic nets, etc.
Inference steps	Dictionary and form-based editor. Transfer tasks are supported.
Inference structures	Dictionary and specialised graphics editor. Inference structures can be decomposed hierarchically.

Support for Visualisation and Development

Visualisation	Inference structures can be displayed using a specialised graphical editor. Inference and role annotations can be displayed in form-based editors which can be accessed with a mouse click. The recommended CommonKADS graphical notation is adhered to throughout.
Development tools	Surprisingly, the libraries of inference structures, inference step types and knowledge role types do not seem to be available within the CommonKADS Workbench, except for a limited selection of inference structures and other types.

C.1.3 Task Knowledge

Coverage and Compliance with CommonKADS

Task Goals	Form-based editor.
Task Bodies	Separate form-based editor.
Task hierarchies	A single task hierarchy, encompassing all top-level tasks in a single application, can be displayed and edited. Each composite task can be decomposed into a separate hierarchy of tasks.

Support for Visualisation and Development

Visualisation	Task hierarchies can be displayed graphically. Decompositions of tasks, and the input & output of particular tasks, can also be displayed in separate windows.
Development tools	The CommonKADS Workbench is the only workbench which fully supports generation of CommonKADS' Conceptual Modelling Language (CML). The workbench includes a utility for transforming CML into FML, the formal modelling language which was developed as a part of CommonKADS [van Harmelen & Balder, 1992]. It is hoped that future developments will include facilities for automatic generation of executable code.

C.1.4 Problem Solving Knowledge

No support is provided for problem-solving knowledge.

C.1.5 Inter-category links

Coverage and Compliance with CommonKADS

Knowledge roles can be "mapped" onto any object in the model ontology or domain knowledge; they can also appear in input/output diagrams at the task level. Inference structures and inference steps can be linked to composite or primitive tasks in the task knowledge.

Support for Visualisation and Development

Creating links between the various categories of knowledge is straightforward using the graphical editors. However, task knowledge editors do not provide access to the knowledge roles at the inference level, so these must be duplicated at the task level in order to create role-task links. The mappings from tasks knowledge to inference knowledge are displayed graphically in the task structure editor; other inter-category links are not displayed graphically.

C.2 Support for knowledge acquisition

C.2.1 Transcript Analysis

Transcript Editing	Facilities are provided for creating transcripts. It is not known whether graphics can be incorporated into transcripts.
Transcript Linking	One or more words in a transcript can be identified as a "fragment", and linked to any basic type from any CommonKADS model. A generic type, *Information*, is also available. Creating links from a transcript to CommonKADS domain knowledge is straightforward.
Ease of Use	Transcripts are easy to annotate (i.e. link to dictionary items or glossary terms). Navigating links between fragments and dictionary items is relatively straightforward; tracing links back requires certain type definitions to be in place, but is otherwise straightforward.

C.2.2 Other KA techniques

No other techniques, e.g. card sorting, repertory grid analysis, are supported.

C.3 Support for other CommonKADS models

Organisational	Individual editors are available for each perspective on the organisational Model. The editors for the Structure and Process perspectives are graphical; the other editors are text-based.
Task	A graphical editor for the Task Model is available within the CommonKADS Workbench.
Agent	A simple graphical editor is available for the Agent model.
Communication	A textual editor is available for the Communication Model. Communication can also be expressed using the input/output windows in the task knowledge within the Expertise Model.
Design	A textual editor is available for the Design Model.

C.4 Maintainability

Object storage	Objects are stored in a central repository; modifications to an object are reflected immediately and automatically in all relevant editors.
Change propagation	Changes made to the names of objects are reflected in the definition of all other objects where it is cross-referenced.
Syntax checking	Checks for naming conflicts and some syntactic errors are performed on all object editors (except on the definitions of an expression and a method body) during model building.
Verification	Explicit validation procedures for completeness and consistency can be invoked for all objects and models.

C.5 Other features

Reliability	The beta version of the CommonKADS Workbench is about as reliable as most beta test versions of software, with occasional minor errors causing interruptions, and one major fault (to do with loading inconsistent versions of files) which caused a crash.
Resistance to misuse	The CommonKADS Workbench is fairly robust; the use of form-based editors minimises the chances of misuse.
Encryption	There are no facilities for encrypting information in parts of the models, e.g. for reasons of security/confidentiality.
Integrity	The project manager can restrict a knowledge engineer's access to certain models. However, there are currently no facilities to prevent a knowledge engineer logging on as a project manager.
Multi-users	The project manager's interface allows the integration of the efforts of multiple knowledge engineers.

C.6 Quality Control

Version control	During evaluation, certain files from a previous version were accidentally overwritten. No facilities for restoring the previous version seem to be available.
Change control	There is no global undo facility.
Modularity	Different knowledge engineers can develop different parts of the knowledge base, which have been segregated by the project manager.
Incremental development	Module management, facilities for export and merging of modules and applications and the automatic updating of cross-references between objects provide support for incremental development.
Documentation	No facilities are known which provide automatic documentation of the above.

C.7 Documentation

The documentation for the CommonKADS Workbench is currently delivered on-line as a set of Postscript documents. It consists of a User Guide, a general guide to the basic features of the workbench, and a guide to demonstrating the workbench.

User guide	Clear and easy to understand, with good indexing.
Basic facilities guide	Not very helpful in day to day use of the system, though good as background material.
Demo guide	Good for initial introduction to the facilities.
On-line help	The workbench only provides a few standard help facilities, but the availability of the manuals in Postscript format means that the manuals themselves can be accessed on-line.

C.8 Costs

The CommonKADS Workbench	£2,000 (UNIX) £1,000 (Windows)
12 months maintenance	£500
CommonKADS Workbench training	Will be provided, possibly by a third party

These prices are valid for 1995, and may well be revised thereafter. A 50% academic discount is available on all prices. Discounts are also available for multiple copies.