Pragmatic KADS: A methodological approach to a small knowledge based systems project

John Kingston

AIAI-TR-110

November 1992

This article first appeared in Expert Systems - The International Journal of Knowledge Engineering, 9, 4, November 1992, pub. Learned Information (Europe) Ltd.

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

Abstract

There has been a growing desire for methodological support for the development of knowledge based systems, and the KADS methodology is probably the most widely known methodology in Europe. However, KADS has been criticised for the overhead which it places on small and medium-sized KBS projects where the risks of KBS development becoming unmanageable are relatively low. This paper describes the use of "pragmatic KADS", an attempt to extract the most useful elements from the KADS methodology, in the development of the COURSE SELECTOR KBS, which assists undergraduate students to choose courses which comply with University regulations and timetabling restrictions. This KBS was built in 6 man weeks. The paper describes the three major phases of a KBS project, giving a brief outline of the KADS approach to each of these stages, and then discusses the techniques which were actually used for COURSE SELECTOR. It concludes with a recommendation of techniques for future small KBS projects.

1 Introduction

As knowledge based systems (KBS) have been applied more widely within the commercial world, there has been a growing dissatisfaction with *ad hoc* methods of KBS development, and a corresponding increasing interest in methodology for developing and constructing KBS. The commercial world wants systems which can be reliably costed, fully validated, easily maintained, and which are amenable to standard management practices. Knowledge engineers want support and direction in KBS development. Previous experience with software engineering methods creates a belief that a methodological approach to the development of KBS could supply these benefits.

There are a considerable number of methodological approaches to KBS, which are reviewed in [6]. In Europe, the KADS methodology is perhaps the most widely known methodological approach to the development of KBS. KADS was developed under the European Community's ESPRIT initiative by a consortium of European partners; it is being further developed in the KADS-II project which is currently under way[1]. Its purpose is twofold: to *formalise* and to *guide* the analysis, design and implementation of knowledge based systems. One of the key features of KADS is its library of *interpretation models*, which are reusable pre-defined frameworks for describing the types of task which KBS perform. It might be thought that KADS, with its specific guidance for the system developer and flow of information for the project manager, would quickly become the standard approach to KBS development.

[1] The deliverables from the KADS-II project, known as CommonKADS, were not available at the time of writing. Some of the expected variations between KADS and CommonKADS are noted in this paper.

There is, however, a vociferous faction amongst KBS developers which believes that methodological approaches to KBS development add so much overhead to some projects that they are not worth using. For example, the KADS methodology has been criticised both for the time required to construct all the detailed models, and for the large number of reports which are required to document progress made and decisions taken. KADS' approach may be essential for large-scale commercial projects, but it is argued that this approach is not appropriate for many KBS developments.

This overhead causes particularly severe problems for small and medium-sized KBS projects. There is also less perceived need for methodology on these projects, since they are typically least at risk from informal KBS development procedures. In order to solve this problem, some KBS developers have rejected methodological approaches altogether; others have developed their own streamlined methodology; and others still have used parts of a recognised methodology, attempting to extract the benefits of formalisation and guidance for developers while minimising document preparation and other overheads. The application of this third approach might be referred to as a *rational* approach [4], or a *pragmatic* approach [15]. [2]

COURSE SELECTOR is a small KBS, developed by the Artificial Intelligence Applications Institute of the University of Edinburgh in 6 man weeks. The use of a full-scale method for such a short project would have been prohibitively time-consuming, and so a pragmatic version of the KADS methodology was used. This paper describes the use of "pragmatic KADS" in the development of the COURSE SELECTOR system, highlighting those parts of KADS which were found to be particularly useful.

2 COURSE SELECTOR: The problem

The COURSE SELECTOR system was implemented for the Department of Business Studies in the University of Edinburgh. The Department's problem was that, in the first two weeks of the Autumn term, every student is required to choose courses for the coming year. Each student has a Director of Studies who is responsible for ensuring that a legitimate combination of courses has been chosen, and every Director of Studies finds that the whole of the first week of term, plus a significant proportion of time thereafter, is taken up with advising students on this complex problem. The task of choosing an acceptable combination of courses is complex, for the following reasons:

- The University of Edinburgh permits students to choose from a very wide range of courses. While most Business Studies students choose their courses

[2] The KADS-II deliverables are expected to include a guide to "configuring" CommomKADS for particular projects.

from the 20 subjects offered within the Faculty of Social Science, it is not unknown for students to take courses such as Chinese Civilisation, or Forensic Medicine. As a result, there are a large number of potential timetable clashes.

- The Department of Business Studies requires students to take certain combinations of courses in their second and third years. All Honours students are required to take Business Studies 1, Business Studies 2 and Business Studies 3 in their first three years; in addition, most students must take six other courses, in one of the following combinations:

  - One subject for three years, and three subjects for one year each
  - Two subjects for two years each and two subjects for one year each
  - One subject for two years, three subjects for one year each, and two extra half-courses in Business Studies

  These regulations can be difficult to coordinate with timetable clashes and students' preferences.

- The Department of Business Studies offers ten different Bachelor degrees which include a Business Studies component. Each of these has different compulsory courses.

- Students are permitted to transfer to Business Studies from other degrees, possibly from other Faculties, and students with appropriate qualifications are permitted to start in 2nd year.

- All the above requirements may be overridden at the discretion of the Head of the Department(s) concerned.

The current procedure (in theory) is for the students to examine the University Calendar, an 800-page volume describing the regulations and timetables of every available course, and to make their course choices which are then verified by their Director of Studies. In practice, many students rely on their Director of Studies to be a source of wisdom, making little or no effort to look at the University Calendar themselves. The result is that the Director has to conduct one or more lengthy interviews with each student. Since each Director is currently responsible for 60 students, the workload is large. There is also considerable scope for error; the number of possible interactions between courses is so great that, during the development of the COURSE SELECTOR system, the University Calendar itself was found to have omitted to mention a timetable clash between two courses which were recommended for a particular degree.

The COURSE SELECTOR system was designed to encode the knowledge stored in the University Calendar, with some additional input from two experienced Directors of Studies. It was used by 2nd and 3rd year undergraduates in October 1991, and it will be used by 1st years as well from 1992.

3 Pragmatic KADS: Analysis phase

The KADS methodology [5] regards the construction of KBS as a modelling process. Once an application for a KBS has been selected[3], a series of models are developed, which gradually transform real-world requirements and expertise into a system implementation. The transformation can be divided into three phases: the *analysis* phase, the *design* phase and the *implementation* phase. The three phases will be discussed in turn; a brief description of the standard KADS methods for each phase will be given, followed by a discussion of the use of pragmatic KADS in the COURSE SELECTOR project.

3.1 The KADS methodology: Analysis phase

KADS provides a great deal of support for the analysis of acquired knowledge. It provides support for knowledge acquisition, in the area of analysing transcripts of interviews, and of structuring the resulting domain information and concepts; it suggests a series of activities for performing initial scoping and feasibility assessment; and it directs developers to investigate the users' requirements and the co-operative roles of the KBS and the user. KADS also specifies a number of reports to be produced to document progress through each of these stages; for example, the requirements analysis should be supported by a report addressing 18 separate issues, which in a large project might become 18 separate reports.

KADS views the process of knowledge acquisition and analysis as a modelling exercise, and so the knowledge engineer is encouraged to structure the acquired knowledge into KADS' four layer *model of expertise*. This model represents problem solving at four levels of abstraction. The lowest level (the *domain* layer) consists of the concepts and relations specific to the domain of the system. The *inference* layer describes the problem solving processes in a declarative manner, and above this the *task* layer provides a procedural interpretation of the inference layer. Finally the top-level *strategy* layer is used to deal with complex problem solving processes involving a number of simpler problem solving tasks.

From the KBS developer's perspective, one of the key contributions which KADS makes to KBS development is the use of *interpretation models*. These models are intended to support the creation of the inference layer of the model of expertise by providing guidance at a high level about the likely inferences in a KBS, and the ordering of those inferences. There are a number of different interpretation models; the choice of an appropriate interpretation model is based on the *task type* being

[3]KADS provides techniques to help with the selection of applications. For example, see [1]

tackled by the KBS. Examples of task types are diagnosis by heuristic classification, planning, assessment, and configuration.

An example of an interpretation model is given in Figure 1:[4]

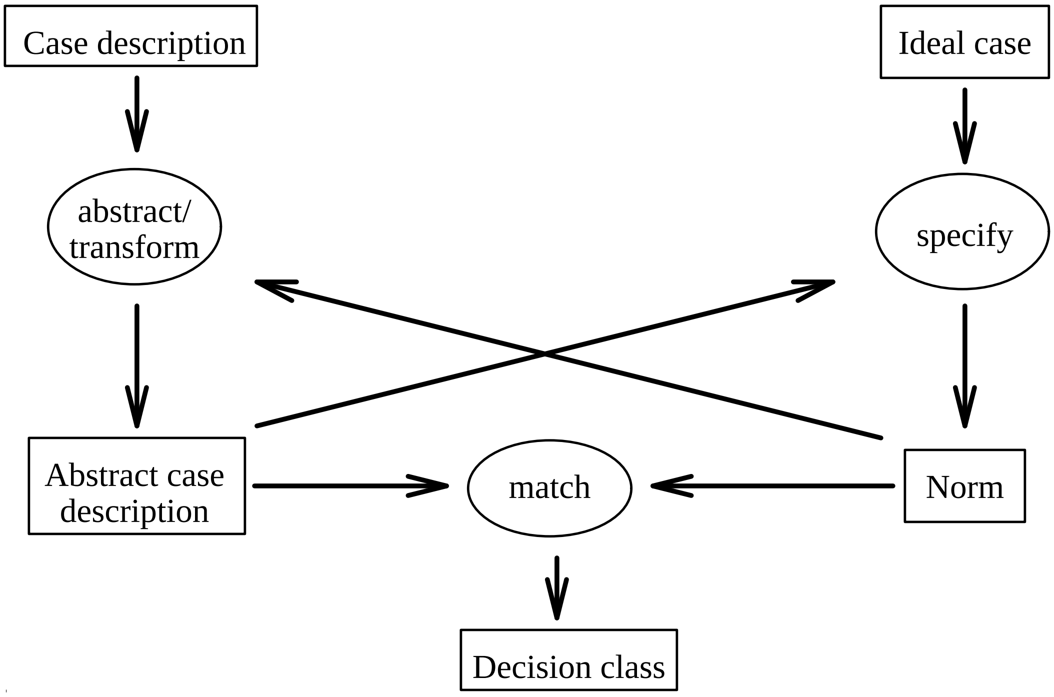

Figure 1: Interpretation model for assessment tasks

This diagram suggests that assessment is performed by comparing key details of a particular case against similar details of an idealised case. For example, if a KBS was being constructed to assess the suitability of applicants for a job, then the applicant's C.V. [case description] might be analysed to draw out key abilities and weaknesses [abstract case description]. These abilities and weaknesses are then compared against the list of key abilities required for the job [norm]; these are based on a real or imagined ideal C.V. [ideal case]. The result of the comparison is the degree to which the key abilities of the candidate match the key abilities required for the job [decision class]. In KADS terminology, the ovals in the diagram represent *inference functions*, which identify the inferences which must be performed at a high level, and the boxes represent *knowledge roles*[5], which identify the inputs and

[4] In pragmatic KADS, this diagram is considered to be an interpretation model, and will be described as such. In KADS, this diagram only shows one component of an interpretation model. This difference is explained further in section 3.3.

[5] The terminology of "inference functions" and "knowledge roles" is based on the terminology currently used in the KADS-II project.

outputs of inference functions, and the type of domain knowledge which will fulfil these roles.

In many problems which are tackled using a KBS, there is no interpretation model which is completely appropriate. In such cases, interpretation models are adapted as necessary. The resulting model may provide valuable insights into the nature of certain expert tasks (for example, see [8]).

3.2 Using KADS for COURSE SELECTOR: Analysis phase

For the construction of the COURSE SELECTOR system, it was decided that the guidance for KBS development provided by the four layer model of expertise was worth the effort required to develop the model. In addition, a low-level analysis of how the KBS and the user would interact with each other was also performed, resulting in a model which is named the *model of interaction*.

The development of the four layers of knowledge for the COURSE SELECTOR system proceeded as follows:

3.2.1 The model of expertise: Domain layer

The vast majority of the domain information was laid out clearly and succinctly in the University Calendar. Only one knowledge acquisition session (plus a few telephone calls) were necessary; these discussions were used to elicit some Department-specific regulations which were not represented in the University Calendar, some knowledge about optional courses which fitted well with certain degrees, and some examples of typical course combinations. As a result, the creation and structuring of the domain layer of knowledge required comparatively little effort.

3.2.2 The model of expertise: Inference layer

The inference layer component of the model of expertise is usually simply referred to as the *inference structure*. The creation of an inference structure involves:

- deciding what type of task the KBS is tackling
- finding the interpretation model for that task type
- adapting and instantiating the inference structure of that interpretation model to the particular domain

It was decided that the task of generating a schedule of courses which fitted in with a range of different restrictions was a *configuration* task. An example of a configuration task is the task tackled by the XCON knowledge based system, where a number of computer components had to be chosen and then correctly placed into boxes [12]. In the COURSE SELECTOR system, a course schedule must be built up from a number of individual courses, some of which are incompatible with each other; the task is therefore analogous to the task performed by XCON, with a course schedule replacing the boxes, and individual courses replacing computer components.

Having decided that the COURSE SELECTOR system was performing a configuration task, the next step was to find the appropriate interpretation model for systems performing configuration. A partial library of interpretation models is available in an early KADS report [3]; however, this library does not include an interpretation model for configuration. Instead, a new "interpretation model" was created, based on a deliverable from the KADS project [16], which describes a case study using KADS to support the development of a KBS for configuring industrial mixers. The derived model is shown in Figure 2.[6]

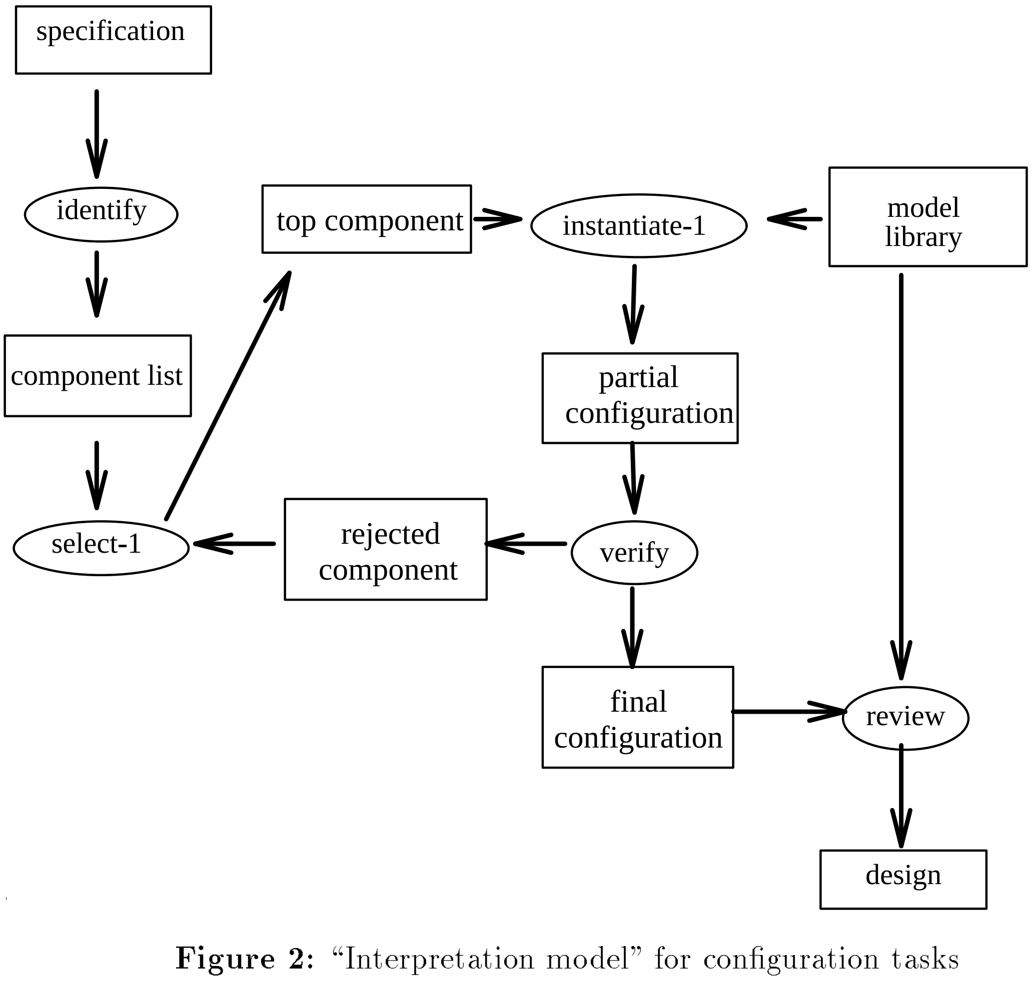

Figure 2: "Interpretation model" for configuration tasks

In the derived interpretation model, a configuration is generated by:

[6]An interpretation model for "incremental configuration" can be found in [5].

1. Producing a list of all components which need to be added to the configuration (the **component list**)

2. Adding components one by one to a partial configuration (represented by the **instantiate-1** inference function)

3. Calculating all the ramifications of that addition, including whether the component will fit into the configuration, and whether the configuration is complete (represented by the **verify** inference function)

4. If the configuration is complete, perform some final actions and produce some output

This interpretation model was adapted for the COURSE SELECTOR system to produce an inference structure, which is shown on the next page. While this inference structure appears to differ considerably from the interpretation model, it follows the same principles:

- All students are required to take certain courses, and these required courses are added to the course schedule first, one at a time. This is represented by the **update-1** inference function, which corresponds to the **instantiate-1** inference function in the interpretation model.

- The area shaded grey represents the course schedule, which corresponds to the **partial configuration** in the interpretation model.

- Students are then allowed to select further courses, a process which is represented by the inference function **select-1**. The selection of a course may make some other courses ineligible, because of timetable clashes or University regulations on the allowed combinations of courses. The alterations to the set of eligible courses are carried out by the inference function **update-2**. As this inference function takes the set of eligible courses as input and produces a new set as output, the inference structure includes an arrow from **currently eligible courses** to **update-2**, even though this "feedback loop" is incorrect syntax according to KADS. This process represents a variation on the interpretation model: courses which will not fit in the timetable are prevented from being selected, whereas the interpretation model suggests that unsuitable components may be selected, only to be rejected later.

- The student continues selecting courses until his schedule is full. The check on whether a student's course schedule is full is represented by **compute** and **compare** on the right hand side of the diagram, which correspond to one of the functions of **verify** in the interpretation model.

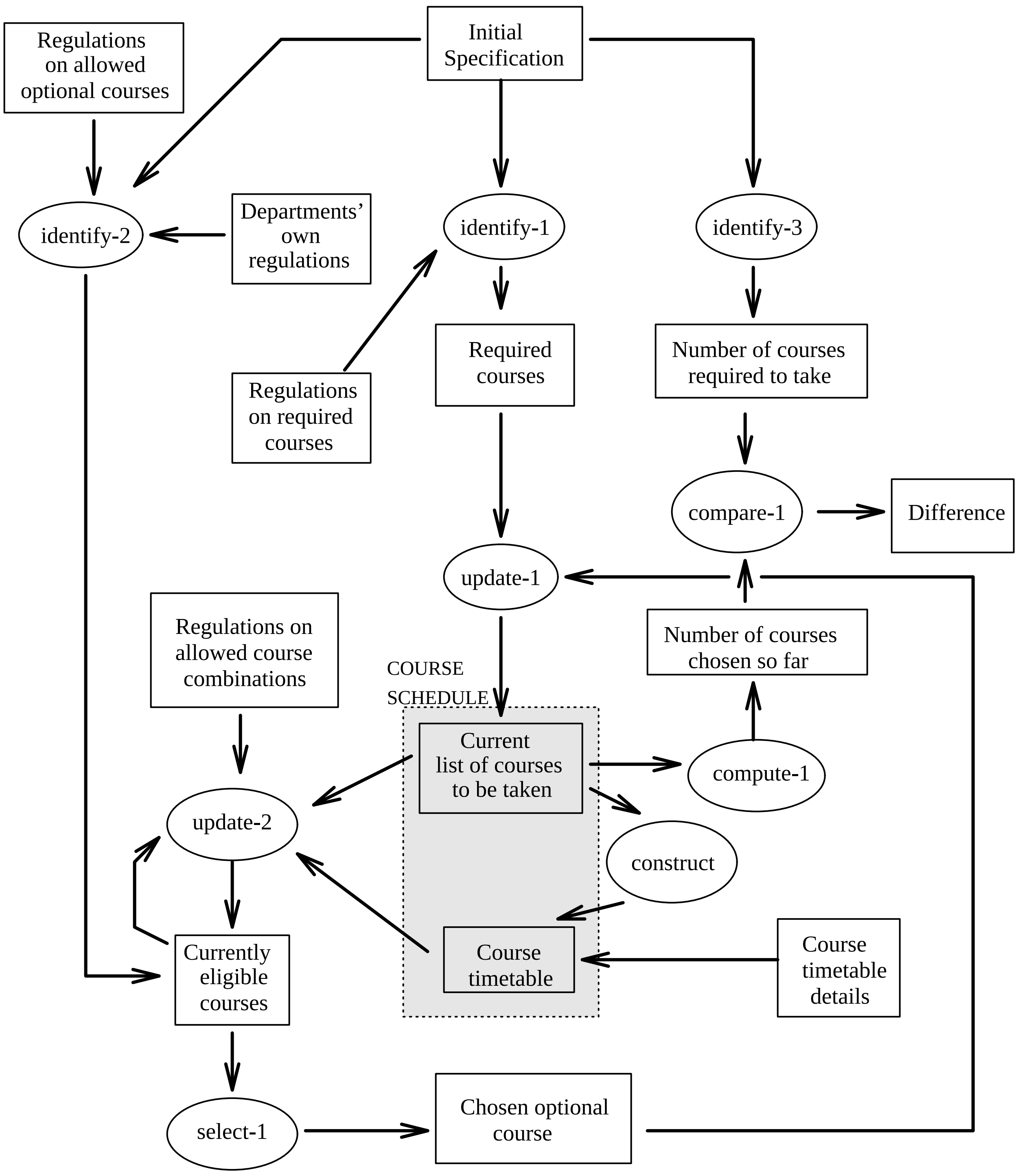

Figure 3: Inference structure for COURSE SELECTOR

Sometimes, certain inference functions are sufficiently complex that they can be decomposed into further inference functions and knowledge roles. This is the case with the **update-2** inference function. The breakdown of this inference function is shown below:

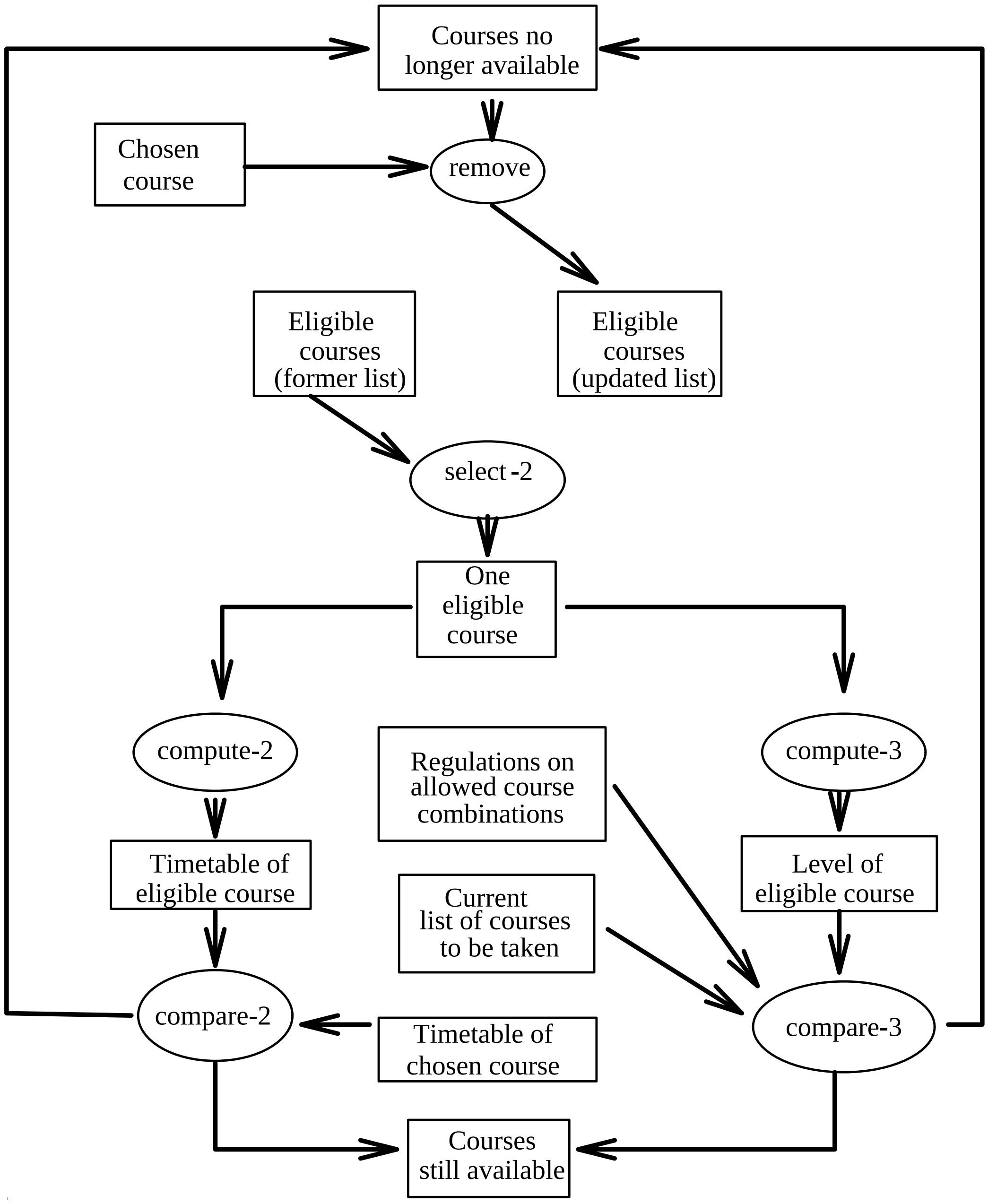

Figure 4: Breakdown of the *update-2* inference structure

3.2.3 The model of expertise: Task layer

Once the inference structure has been developed, then a *task structure* should be developed to determine the order in which inferences should be carried out. This task structure provides the task layer of the model of expertise. The task structure of the COURSE SELECTOR system is shown below:[7]

task *configure-course-schedule*
goal assist a student to choose a set of courses which comply with University regulations
control-terms
Required-courses → set of all courses which the student is required to take
Eligible-courses → set of all courses which the student is permitted to take
task structure
configure-course-schedule(Required-courses + Eligible-courses → course-schedule)
obtain(initial information)
identify(Required-courses)
identify(Eligible-courses)
identify(number of courses to be taken)
do for each(required-course in Required-courses)
update-1(current-course-schedule → current-course-schedule including required-course)
end
while (number of courses in current-course-schedule < number of courses to be taken) do
compare(number of courses in current-course-schedule with number of courses to be taken)
select-1(optional-course from Eligible-courses)
update-1(current-course-schedule → current-course-schedule including optional-course)
compute(number of courses chosen)
update-2(Eligible-courses)
end
assign(current-course-schedule → course-schedule)

The primary purpose of the task structure in the COURSE SELECTOR project was to specify a procedural ordering of operations - both inference functions from the inference structure and input/output functions. For example, the comparison of the number of courses chosen against the number required to be taken was placed at the beginning of the **while** loop, because it is possible that the student's course

[7] The **update-2** inference function has not been broken down in this task structure, for the sake of brevity.

schedule may be filled by the required courses, in which case the selection of optional courses will not be needed.

3.2.4 The model of expertise: Strategy layer

The Strategy layer is concerned with decisions regarding complex problem solving, such as the resolution of possible deadlocks, or the sequencing of potentially conflicting tasks. There are few KBS projects for which a strategy level analysis is necessary; on the COURSE SELECTOR project, it was not needed.[8]

3.2.5 The model of interaction

The other model which was developed for the analysis phase was a *model of interaction*, which is shown in Figure 5. This model is not part of the KADS methodology; it was derived from KADS' model of cooperation [1], which models the distribution of labour *between* problem-solving tasks. The model of interaction uses similar techniques in order to model the distribution of labour *within* a particular problem-solving task [9]. Its purpose is to model the interaction between a KBS and a user, particularly in deciding whether each operation should be performed by the KBS, the user, or the system and user working together. While KADS expects these decisions to be modelled in the task structure, the model of interaction explicitly represents the fact that one inference structure can be open to multiple modes of interaction. For example, a KBS may allow users either to volunteer information or to be prompted for the same information. Many users of KADS associate an inference structure with a single task structure, and by implication with a single mode of interaction; an explicit model of interaction helps to clarify decisions about modes of interaction, as well as identifying any omissions from the task structure.

The model of interaction is created by

- drawing the task structure in diagrammatic form
- identifying dependencies between tasks and identifying external sources of information
- assigning tasks to either the KBS, the user, or the two working together.

The final model for the COURSE SELECTOR system is shown on the next page. It suggests that almost all the tasks will be performed by the knowledge based system alone; only two tasks (gathering initial information and selecting a course) are performed by the system and the student working together. It also highlights certain information which needs to be accessed by the KBS via file parsing, or integration with other software packages.

[8] The KADS-II project is considering treating the strategy layer as a separate model, rather than a layer of the model of expertise.

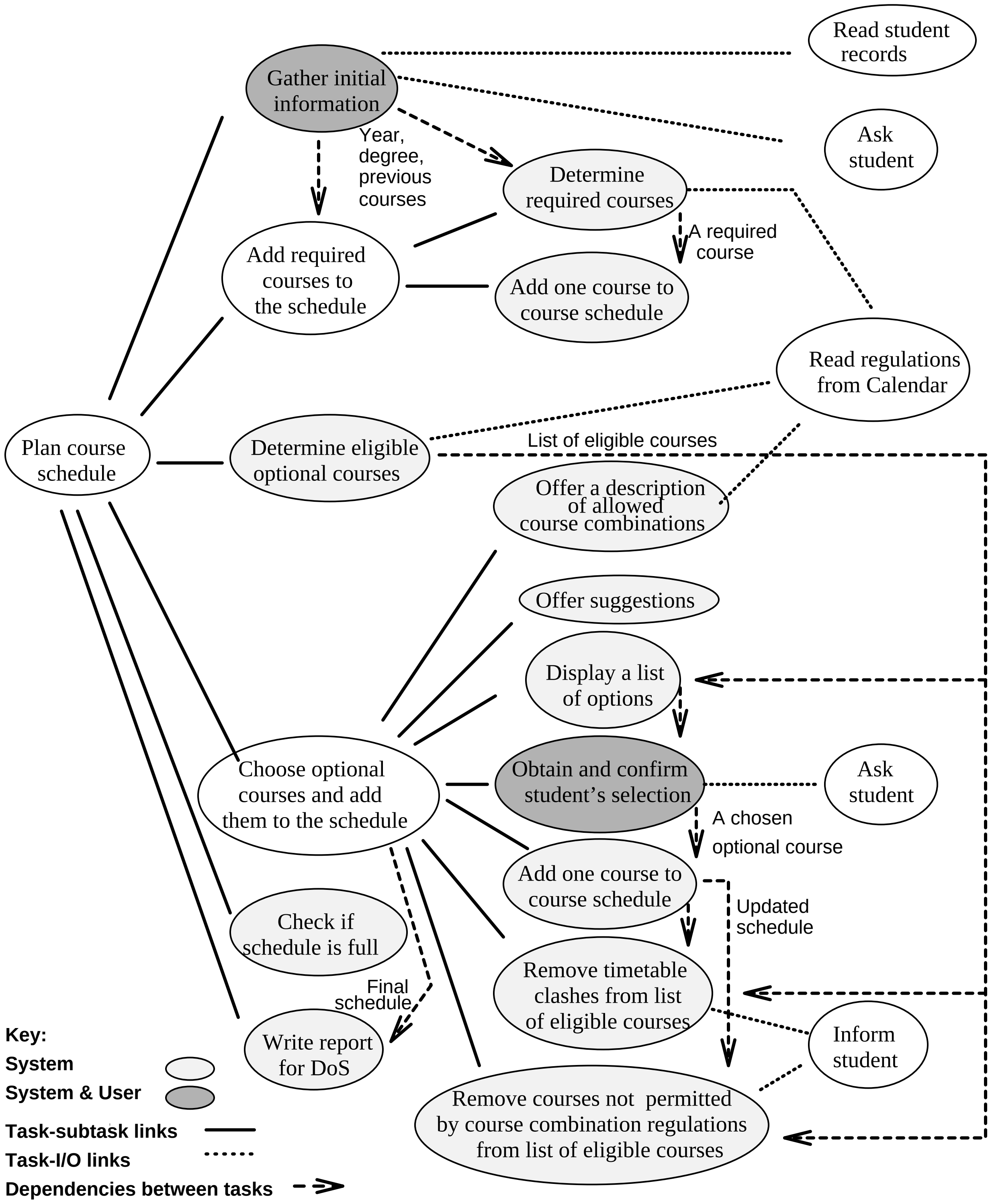

Figure 5: The Model of Interaction

3.3 Differences between KADS and pragmatic KADS: Analysis phase

The models developed in the model of expertise for the COURSE SELECTOR project differ slightly from those recommended by KADS. These differences are as follows:

- In KADS, interpretation models include two components: a generic inference structure and a generic task structure. Pragmatic KADS currently uses only the inference component of the interpretation model. [9]
- In KADS, the inference layer of the four-layer model of expertise represents the dependencies between the inferences which an expert performs, and the knowledge which is used in these inferences. In pragmatic KADS, the inference structure represents every action which the expert takes, rather than just the dependencies between inference and knowledge. For example, the inference structure shown in Figure 3 includes a comparison of the number of courses a student is required to take with the number of courses taken so far. This comparison is a task which a Director of Studies is required to perform. However, the only function of this comparison is to decide when a course schedule is full, and KADS places all control functionality in the task structure rather than the inference structure. Yet this comparison is a vital part of the overall problem-solving procedure, so pragmatic KADS includes it in both the inference structure and the task structure, for the sake of clarity.
- The differences in pragmatic KADS' inference structure have a knock-on effect on the task structure. In KADS, the task structure is used to decide and fully specify flow of control and input/output, as well as specifying control tasks such as checking if a course schedule is full. In pragmatic KADS, the primary purpose of the task structure is to decide on the flow of control, which consists of specifying an ordering on tasks, and identifying any iteration or recursion. Input and output are also identified and assigned in the task structure, with the assistance of the model of interaction.

The net effect of these differences is to bring the results of the analysis phase of pragmatic KADS closer to the design and eventual implementation of the KBS than the more abstract models recommended by KADS.

[9] It is possible that the generic task structures may be useful in some "pragmatic" projects, particularly at the stage of selecting an appropriate interpretation model.

4 Pragmatic KADS: Design phase

The design phase is the second major phase of the KADS methodology. As for the analysis phase, the KADS approach to design will be outlined briefly, after which the approach actually taken in the COURSE SELECTOR system will be described.

4.1 The KADS methodology: Design phase

The suggested process for KBS design in KADS is similar to the design processes suggested by many conventional methodologies. It involves performing a functional decomposition, and implementing the resulting individual functions. It is a three-stage process:

- **Functional design**: decomposing the models produced in the Analysis phase into groups of problem solving functions, storage functions and cooperation functions
- **Behavioural design**: selecting AI "design methods", such as data-driven reasoning, model-based reasoning or best-first search, to implement each function.
- **Physical design**: selecting rules, objects, formulae, or other data structures to implement the chosen design methods, and arranging the chosen data structures into modules in a sensible fashion.

The key to successful design using KADS is to preserve the structure of the model of expertise, particularly the inference structure, in the design. To this end, KADS guides the knowledge engineer to translate inference functions into "problem solving functions", and knowledge roles into "storage functions". The model of interaction integrates well with the KADS approach to design by making it easy to produce "cooperation functions". The functional decomposition consists of these three sets of functions, plus an indication of the data flow and control hierarchy.

Both functional design and physical design are techniques taken almost directly from common software engineering practice. It is the middle stage, behavioural design, which is most specific to KBS. Unfortunately, KADS offers little guidance on design method selection. What guidance there is can be found in [2], but the suggestions provided are not very detailed, and there is no guidance at all for configuration tasks[10]. Nor does KADS give any guidance on the available features in KBS tools, although this is understandable because the market for KBS tools changes so rapidly. The reader is referred to recent reviews such as [7], [14] and [13] for such information.

[10]It is hoped that the KADS-II project will provide much more guidance on design, possibly including a "design model library", analagous to the library of interpretation models for the analysis phase.

4.2 Using KADS for COURSE SELECTOR: Design phase

4.2.1 Functional decomposition

A functional decomposition was performed for the COURSE SELECTOR project, but it was not found to be particularly useful. There were two main reasons for this:

- Functional decomposition in KADS is the first stage in making the analysed knowledge less abstract. Since the model of expertise in pragmatic KADS is more concrete than the KADS equivalent, functional decomposition has less of a role.
- Pragmatic KADS does not use KADS techniques for behavioural design, because of the lack of guidance on design method selection. The technique which was used makes use of many different sources of input, with the functional decomposition being a fairly minor contributor, instead of being almost the sole contributor.

The functional decomposition did prove useful in identifying a few communication paths and minor knowledge roles which had been omitted from, or not fully specified in, the models of expertise and interaction.

4.2.2 Behavioural design

Because KADS provides almost no guidance on behavioural design[11], a different approach was used for the COURSE SELECTOR project. The selection of design methods was based on an approach known as the "probing questions" approach, originally proposed by Kline and Dolins [10] and developed further at AIAI [11].

This approach requires the system designer to ask himself a number of questions about the system. The answers to the questions may arise from user requirements, any layer of the model of expertise, or the data flow specified in the functional decomposition. Certain answers will suggest the use of certain design methods. Examples of probing questions include:

- Question: "Will the KBS have to reason about relations between things?"

  Elaboration: "Many KBS do such reasoning, about connections between components, things which are part of other things, or dependencies between events and decisions"

  Recommendation: "Consider a representation system which supports *user-defined relationships*"

[11] The only guidance known is to be found in a KADS project report [2].

- Question: "Will the KBS reason with incomplete data?"

  Elaboration: "In deciding whether this is true of any particular system, one must distinguish missing data (situations where the data available to the system varies from case to case) from situations where the results which the system may determine are unknown"

  Recommendation: "If the system has to cope with missing data, then a number of techniques may be applicable:"

  - *Data-driven reasoning*
  - *Certainty factors*
  - *Default values*
  - *Truth maintenance systems*

For the COURSE SELECTOR system, two "probing questions" led to suggestion of design methods. The first question asked whether it was sensible to enumerate all possible solutions to the problem, or whether the system should be capable of generating solutions. In a configuration problem, there are a very large number of possible solutions, and so it is better that solutions are generated. *Data-driven reasoning* is therefore suggested. It is also likely that the partial configuration will need to be explicitly represented, which suggests the use of *objects* and *dynamic object creation*. The second question asked whether the system has to re-make elaborate decisions; the answer is yes, if the user should decide to undo a choice. The consequent suggestion is to use either *backtracking* or *truth maintenance*.

In summary, the probing questions analysis recommended the use of data-driven reasoning and truth maintenance. It also suggested that objects, with dynamic object creation, might be useful as well.

Design method selection is necessary not only for problem solving functions, but also for representing storage functions (domain information). An analysis of the maintenance requirements for the COURSE SELECTOR system had determined that the characteristics of courses are likely to change frequently (every year, at least). It was therefore decided that the information about courses would be read from a file, rather than being hard-coded into the KBS.

4.2.3 Physical design

The physical design of a knowledge base which has been designed using KADS involves deciding how to implement the chosen design methods. Ideally, the first stage would be the selection of a tool which supports all the desired design methods. However, in the COURSE SELECTOR project, as in many projects, there were a number of other factors affecting the choice of tool apart from the design method selection. The principal factor was that the COURSE SELECTOR system should be able to run on an 8086 PC with 640KB of RAM, and that the tool should cost very little.

The tool chosen was CLIPS version 5.0. CLIPS is a KBS toolkit whose primary form of representation is forward chaining rules[12] , which are similar to the rule-based component of ART-IM. It includes a simple truth maintenance system, and dynamic creation of symbolic facts.

A brief description of the physical design used for each of the design methods is given below:

- Data driven reasoning was implemented using forward chaining rules.
- The suggested object-oriented representation was actually implemented using facts, where each object was represented by several facts. The first element in each fact was the name of an "object".
- Truth maintenance was also implemented using facts, since the built-in truth maintenance facility was not sufficiently expressive. Truth maintenance was implemented by the simple but powerful technique of creating a fact to represent a course which was known **not** to be eligible for selection. This contrasts with the normal truth maintenance technique of keeping track of **valid** assumptions; the reason for this choice was that there are likely to be fewer ineligible courses than eligible ones, and so fewer "truth maintenance" facts will be required, leading to increased efficiency. The "truth maintenance" facts note the reason for the creation of the fact, which will be the addition of a certain course to the course schedule; if that course is ever removed from the schedule, then any "truth maintenance" facts associated with it are also removed. This technique is powerful because it is able to handle a situation where a course is ineligible for more than one reason; a course is only considered eligible if *all* the "truth maintenance" facts affecting it are removed.
- The external file of course information was developed by using a spreadsheet, and writing out a text file containing the fields of the spreadsheet. This file was then parsed using an ASCII parser.

In addition, it was decided that the KBS should be broken down into separate files of rules, and that the content of these files should mirror the functional decomposition (and hence the models of expertise and interaction) as far as possible. This decision helped in the debugging of the KBS, and clarified later decisions about where to store certain rules.

[12]Version 5.0 of CLIPS also includes object-oriented programming and functions, but these were not compiled into the version of CLIPS which was used, because of restrictions of speed and memory on the delivery PCs.

5 The KADS methodology: Implementation phase

The implementation phase is the third major phase of a KBS project; however, KADS has very little to say about it. Once the physical design has been completed, implementation is seen as a programming task, where software engineering methods can be applied with little adaptation.

The full details of the implementation of the COURSE SELECTOR system are likely to be of interest to CLIPS programmers only, and are therefore not described here. Since KADS' only guidance on implementation is to use software engineering techniques, there are no pragmatic KADS methods for implementation.

6 Evaluation and Recommendation

The analysis and design phases of pragmatic KADS for the COURSE SELECTOR project took approximately 5 man days to complete. The major benefits of using pragmatic KADS for the COURSE SELECTOR project were as follows:

- The development of an inference structure. The interpretation model were found to be very useful as a basis for structuring the knowledge, and also for providing a structure for the final implemented system.
- The task structure was necessary to complement the inference structure.
- The model of interaction was useful in identifying I/O functionality, and as a check on the completeness of the task structure. In other projects, it may be even more useful if it identifies certain tasks that should be carried out by the user without any assistance from the KBS, since these tasks can then be omitted from the design of the KBS [9].

The guidance provided by KADS on the design phase is not particularly useful in its current form. The use of pragmatic KADS, where the model of expertise is more closely related to the design than in KADS, made the functional decomposition almost superfluous. The "probing questions" approach to behavioural design was found to be useful both at a control level and at a more detailed level. KADS' emphasis on structure-preserving design was considered to be important, both for current development and future maintenance.

The recommendation for future small projects is to use a KADS interpretation model as a starting point for structuring the acquired knowledge, and to develop a model of expertise. The inference layer of the model of expertise should model what the expert actually does, not just dependencies between knowledge and inferences. A model of interaction should also be developed, to clarify the I/O of the system, refine the task structure, and perhap identify tasks which can be omitted from the KBS design.

Once the analysis phase is completed, the design phase should preserve the structure of the models of expertise and interaction in the KBS design. A functional decomposition may be helpful here, but is not essential. Probing questions provide useful heuristics for making decisions about design methods.

Acknowledgements

I would like to acknowledge the contributions of Gail Anderson and Ian Filby of AIAI, and David Porter of Touche Ross Management Consultants for their comments on drafts of this paper. I would also like to acknowledge Dr. Robert Inder of the Human Communications Research Centre, University of Edinburgh, for suggestions on the implementation of truth maintenance functionality.

References

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

[2] G. Schreiber (ed). *A KADS approach to KBS design*. University of Amsterdam, 1989. ESPRIT project 1098, Deliverable B6, UvA-B6-PR-010.

[3] J.A. Breuker (ed). *Model-driven Knowledge Acquisition*. University of Amsterdam and STL, 1987. ESPRIT project 1098, Deliverable A1.

[4] D.S.W. Tansley & V.A.M Farrell. Requirement Analysis of KATE using KADS. In *Proceedings of BCS Methodologies Workshop*, 3-4 December 1991.

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

[6] R. Inder and I. Filby. Survey of Knowledge Engineering Methods and Supporting Tools. In *Proceedings of BCS Methodologies Workshop*, 3-4 December 1991.

[7] J. Kingston. Building Knowledge Systems: Toolkits and Case Studies. In *Quantitative Methods, Supercomputers and AI in Finance*. Unicom Seminars, Brunel Science Park, Cleveland Road, Uxbridge, Middlesex, 5-6 Feb 1992. Also available from AIAI as AIAI-TR-102.

[8] J.K.C. Kingston. X-MATE: Creating an interpretation model for credit risk assessment. In *Expert Systems 91*. British Computer Society, Cambridge University Press, 17-18 Sep 1991. Also available from AIAI as AIAI-TR-98.

[9] J.K.C. Kingston. The model of interaction. *Newsletter of the BCS Methodologies Interest Group*, (1), August 1992. Also available from AIAI as AIAI-TR-115.

[10] P. Kline and S. Dolins. *Designing Expert Systems: A guide to selecting implementation techniques.* John Wiley & Sons, 1989.

[11] C. MacNee and J. Kingston. PDQ: A KBS to help KBS designers select knowledge representation and inference techniques, 1993. *Submitted to the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, 1-4 June 1993.* Also available from AIAI as AIAI-PR-54.

[12] J. McDermott. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19, 1:39–88, 1982.

[13] W. Mettrey. Expert Systems and Tools: Myths and Realities. *IEEE Expert*, pages 4–12, February 1992.

[14] C. Price. *Knowledge Engineering Toolkits.* Ellis Horwood, 1991.

[15] D.S.W. Tansley. Personal communication, 1991.

[16] J. Wielemaker and J-P. Billault. *A KADS analysis for configuration.* University of Amsterdam, 1988. ESPRIT project 1098, Deliverable F5, UvA-F5-PR-001.