Knowledge Engineering for Practical Applications

M Mulvenna, P Ye, M McTear[1],M Murphy[2]
NIKEL, University of Ulster, Newtownabbey,
Co. Antrim, UK BT37 0QB
email: {md.mulvenna,p.ye,m.murphy}@ulst.ac.uk
tel: +44 232 365131 x 3166
fax: +44 232 362803

Keywords: Knowledge Engineering, Knowledge Based Systems

1. Introduction

A method is necessary to develop any piece of complex software. There are many methodologies which span the spectrum of software development. These range from specialised methods which accommodate real-time issues, to methods which embrace object-oriented systems. In addition, for Knowledge Based System (KBS) development, the pre-eminent methodology is KADS[1].

The argument advanced in this paper is that, when building KBS for practical applications, the method used for analysis, design and development may not necessarily be a specialised KBS method. There are reasons for this decision. Firstly, the new generation of KBS form only a part of an IT solution. That is, the KBS *may not* be the dominant IT component. Therefore any method must encompass the whole IT solution. Secondly, as will be shown by the case studies outlined in this paper, the term "KBS" now encompasses many disparate AI technologies. Specialised KBS methodologies may be restrictive for the development of systems utilising Case Based Reasoning (CBR), Constraint Logic Programming (CLP), Fuzzy Systems, Adaptive Neural Networks, Object-Oriented Systems (OOS), or their combinations.

2. Software Engineering for Knowledge Based Systems

Practical experience in building Information Systems (IS) with a KBS component has resulted in the advancement and evolution of a method which can minimise possible pitfalls of such development (Figure 1). The method is a form of the standard software life-cycle model (LCM), with a heavy dependency on object-oriented components. In addition, to facilitate KBS design inclusion in the process, protoyping and knowledge elicitation are significant phases.

Figure 1 is an informal representation of the method. The method is not as reliant on the classical waterfall model as it may seem. For instance, the use of object-oriented modelling at the analysis and design phases results in the coalescing (to an extent) of the two phases. At all stages, internal evaluation and review processes skew the method towards iterative development.

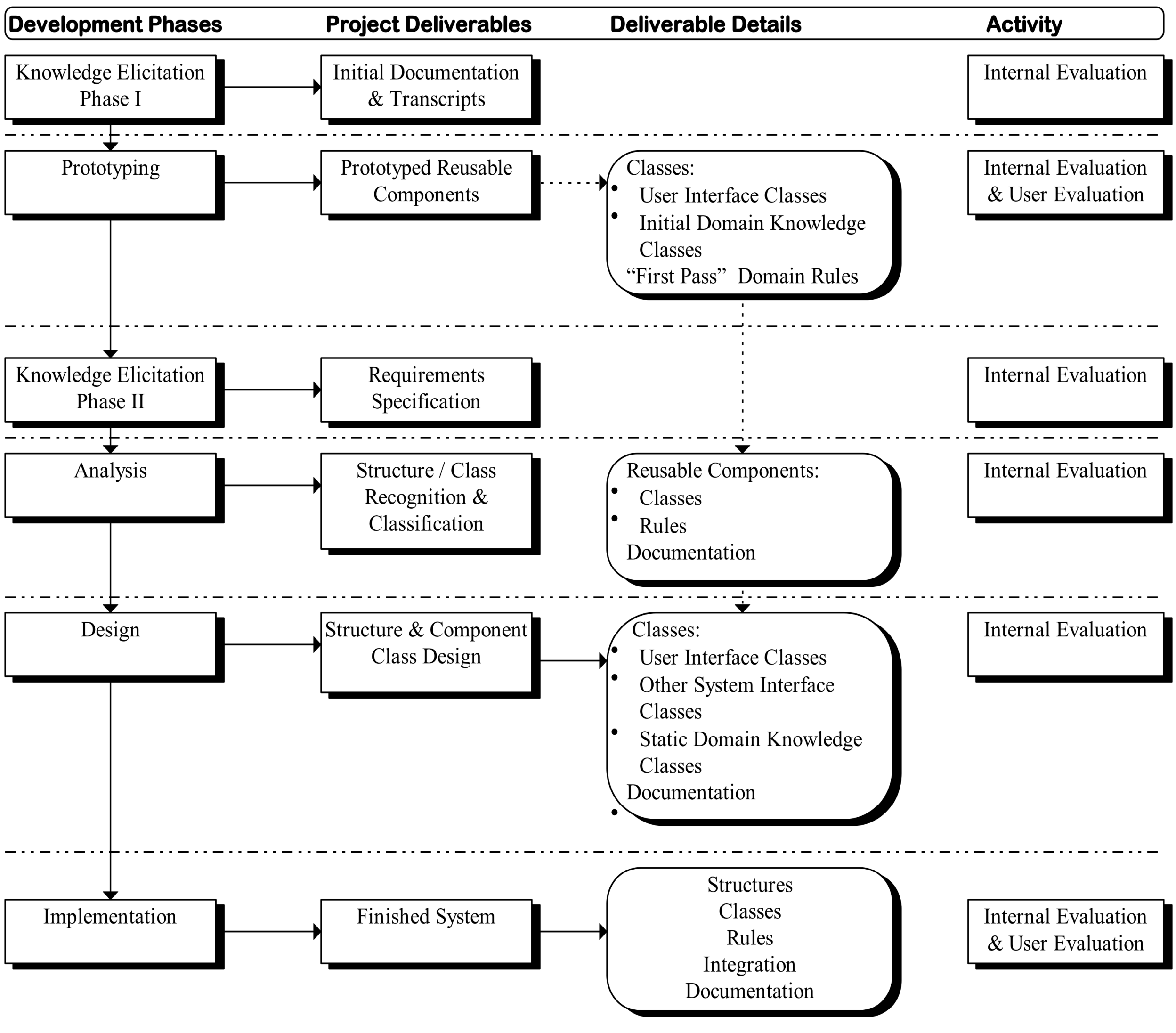
The first development phase, *Knowledge Elicitation Phase I*, encompasses the initial interviews and discussions which provide an awareness of the problem to be solved for the developers. All people involved in the project, from end-users to senior management are interviewed. The scope of the initial interviews is broad, with issues such as system acceptability, existing IS infrastructure mergers, and the context of the problem all being examined. Eventually the interviewing process becomes more focused on the problem domain, with longer, in-depth interviews taking place.

The second development phase, *Prototyping*, has an important function. It provides feedback to the developer on the accuracy and completeness of the prototyped system. Issues such as usability and correctness of the system are closely examined, in conjunction with the domain knowledge sources and the end-users. When prototype development is carried out with an object-oriented development environment, the prototype may also generate reusable software components. Many of these components are class definitions for the user interface and also for the static domain knowledge. Crude "first pass" rules may be scoped at this stage.

Knowledge Elicitation Phase II is a more formal elicitation phase. The deliverable from this phase is a System Requirements Specification (SRS). This document is used as the basis for contract of the full system development.

[1] Institut für Computerlinguistik, Fachbereich Informatik, Rheinau 1, Universität Koblenz-Landau, 5400 Koblenz, GERMANY

[2] Department of Information Systems, University of Ulster, Newtownabbey, Co. Antrim, UK BT37 0QB,
Email: mctear@informatik.uni-koblenz.de



Development Phases
Project Deliverables
Deliverable Details
Activity
Knowledge Elicitation Phase I
Initial Documentation & Transcripts
Internal Evaluation
Prototyping
Prototyped Reusable Components
Classes:
User Interface Classes
Initial Domain Knowledge Classes
"First Pass" Domain Rules
Internal Evaluation & User Evaluation
Knowledge Elicitation Phase II
Requirements Specification
Internal Evaluation
Analysis
Structure / Class Recognition & Classification
Reusable Components:
Classes
Rules
Documentation
Internal Evaluation
Design
Structure & Component Class Design
Classes:
User Interface Classes
Other System Interface Classes
Static Domain Knowledge Classes
Documentation
Internal Evaluation
Implementation
Finished System
Structures
Classes
Rules
Integration
Documentation
Internal Evaluation & User Evaluation

Figure 1. Prototyping & Knowledge Elicitation Using Object-Oriented Method

The *Analysis* phase concentrates on the identification and classification of object and structures. Any of the surviving components from the *Prototyping* phase may be reused at this stage. Much iteration will go into the modelling of classes and structures at this stage.

In the *Design* phase, decisions are made on the formal object specification. This entails the design of all domain knowledge classes, all user interface classes, all ancillary structures (such as DBMS links), and formal documentation generation.

Finally, the *Implementation* phase integrates the object design in a programming environment.

3. Case Studies

3.1 Case Study One: Product Specification & Shop Floor Scheduling System

In common with many manufacturing operations, the company in this case study has two major areas of concern:

- Ensuring customer satisfaction;
- Maximising earnings by minimising cost.

The two areas are intimately interlinked. Firstly, to ensure that the customer is satisfied, these goals must be achieved:

- Rapid response to customer enquiries;
- Providing a price to the customer that is competitive;

- Providing accurate product specifications.

Secondly, to minimise costs, the following goals must be reached:

- Accurate costings;
- Efficient loading of shop-floor machinery;
- Flexible loading of shop-floor machinery.

A system that provides accurate product specifications, and schedules resultant customer orders on the shop-floor, would address the concerns of customer satisfaction, and result in lower manufacturing costs. Both product specification and scheduling are dealt with in the following sections.

3.1.1 Product Specification

In order to generate accurate product specifications, the sales, customer support, and shop floor personnel in a commercial organisation must be aware of the various constraints affecting their business function. Essentially, sales people are motivated by the financial reward that results from a sale, while production engineers and management attempt to produce the goods at lowest costs. A system that attempts to bridge the information gap between these two groups of people in a manufacturing organisation must combine precise engineering specifications with more vague 'rules of thumb'.

Because of the variety, complexity, and limited availability of this information, inaccurate quotes may be passed to the customer. With the use of a Knowledge Based System (KBS) that combines the engineering specifications with the rules governing the quoting/ordering process, optimally costed product specifications can be generated.

3.1.2 Scheduling

Scheduling involves assigning times and resources to activities so that a set of constraints is best satisfied according to a set of pre-defined objectives. It is a difficult problem and much of the difficulty comes from the need to schedule a large number of activities, and to attend to a diverse set of objectives, requirements, and constraints which are often in conflict with each other. A good schedule must reflect *a satisfactory compromise* among these competing influences.

Because of the amount of information involved and the complexity of the scheduling process, human schedulers typically do not handle them well. They usually employ simple rules and consider limited information to approach a (or any) solution which is often of merely short-term benefit. With Knowledge-Based Systems (KBS) and other information techniques, a optimal solution can be reached by a knowledge based scheduling system using:

- data and information concerned with jobs and the shop floor;
- knowledge concerned with scheduling strategies and the rules of thumb.

3.1.3 Use of Knowledge Engineering Method

The following diagram (Figure 2) provides information on the architecture of the product specification and scheduling system. The product specification component provides an output which is read directly into the company's MIS system. The MIS then acts as a database to the scheduling component of the system.

Development work for this system is on-going. At a very early stage in knowledge elicitation it was decided that the existing MIS system would be used as a data repository. This greatly simplified the data management aspect of the system. The product specification component of the system has been prototyped using a high-level object-oriented development tool. Most of the interface classes, and many of the static domain knowledge classes will be reused in the full development.

The scheduling software is a Constraint Logic Programming (CLP) language [2, 3] , based on Prolog. A throwaway prototype has been developed in this language to test the efficiency of some of the mathematical models devised to manage the heuristics of the scheduling process.

3.1.4 *Benefits of System*

The following benefits are realised by the development of the system. From the product specification component, it will :-

- Increase customer satisfaction by providing instant product feasibility verification at the quotation stage in conjunction with accurate, standardised and consistent quotes;

- Free employees' time;
- Make scarce information widely available throughout the company;
- Preserve knowledge within the company over a long period of time;
- Facilitate easier training of Salespeople and Product Engineers;
- Ensure that the best, most cost-efficient routes would be calculated every time for an order thereby ensuring the most competitive prices possible;
- Eliminate mistakes on costing.

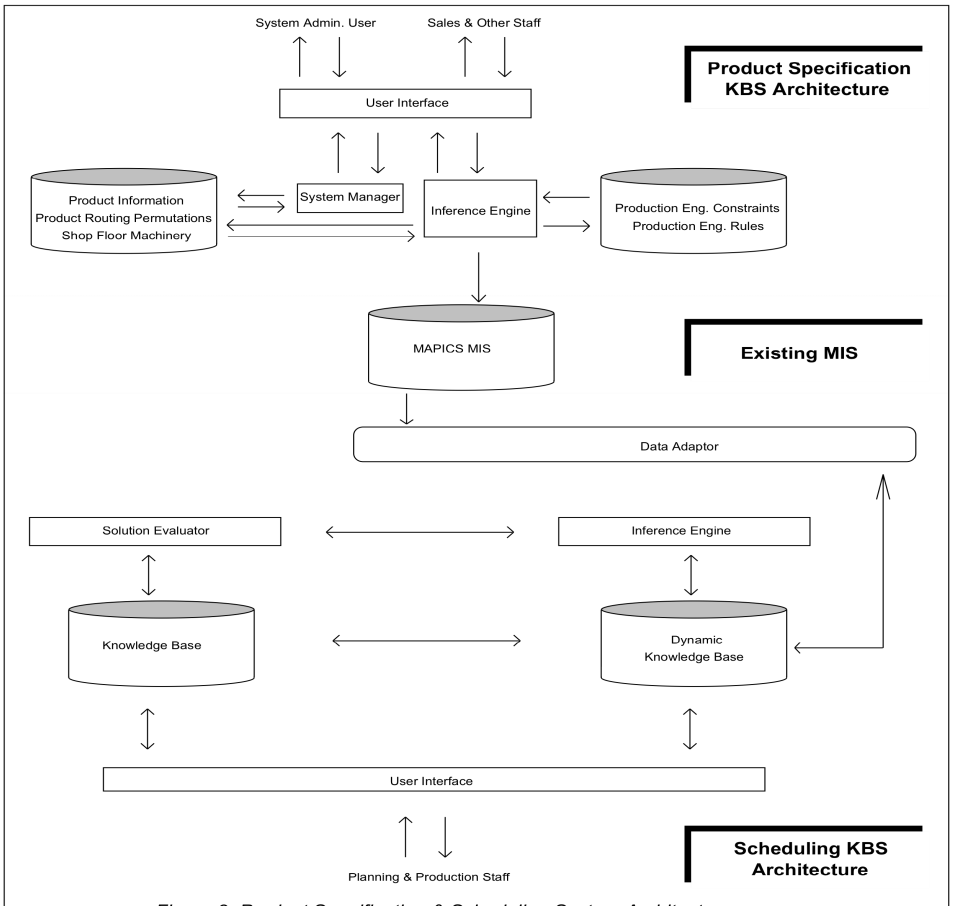


Figure 2. Product Specification & Scheduling System Architecture

The scheduling component will bring the following benefits:-

- Better control over global optimal planning;
- (resulting in) Better attainment of customer delivery dates;
- Realistic real-time scheduling;
- (leading to) Better forecasting and decision making.

3.2 Case Study Two: Product Support System

The development of a state-of-the-art product support system for a major aerospace manufacturer is the subject of the second case study. The system incorporates leading edge computing technologies to advise and assist aircraft support engineers in making accurate assessments of reported faults.

3.2.1 Product Support

Product support engineers in this company have many years of experience in building and maintaining aircraft and aircraft parts. They are particularly adept at fault-finding and trouble shooting. When a fault occurs in an aircraft used by an operator company, the product support engineer uses all available resources to identify and solve the problem. These resources include his own experience, his colleagues' experience, and all the past occurrences of faults, which are on paper file. In addition, there are many revisions of product support manuals available for study. The current system in place at the company is a paper-based one, where incoming faults notices are receipted and routed to available engineers.

3.2.3 Use of Knowledge Engineering Method

Figure 3 provides information on the architecture of the product support system. The core of the system is a relational DBMS which provides the standard "help desk" functions of call logging, routing, managing, tracing, reporting, etc. A case base repository is coupled with the DBMS. The product support staff enter new faults that are stored in the DBMS, and then matched against all cases stored in the case base. The retrieved case(s) are then presented to the user, with backup material from the digitised product support manuals available on-line.

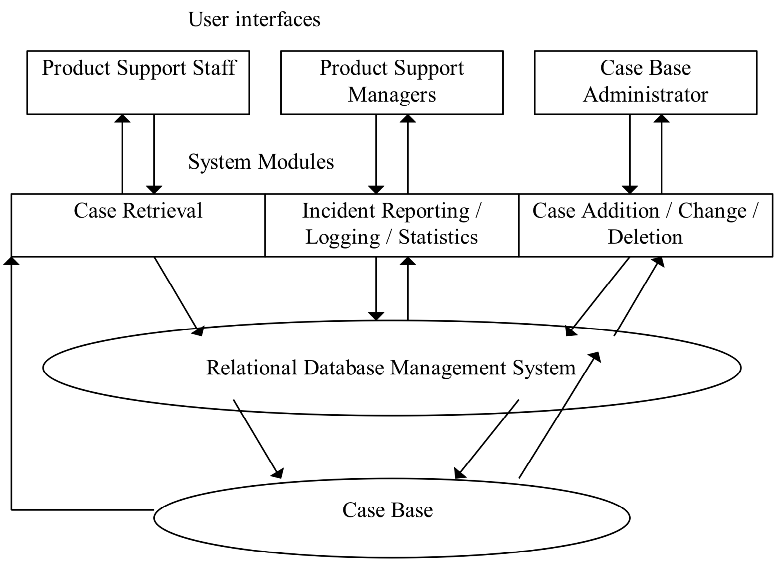


Figure 3. Product Support System Architecture

The role of the system is to assist the engineer in his decision-making process, and to provide and enhance the flexibility that was available with the paper-based system. A prototyped version, which exhibited the case-based reasoning component of the system was developed for assessment by the product support engineers. At this stage the proposed functionality of the full system was mimicked in the user interface. This allowed the engineers to partially assess the usability of the finished system. Much of the low level RDBMS/CBR interfacing and the user interface was retained from the prototype for reuse in the full system. An additional requirement for this system was that the software design would be as generic as possible. That is, the system could be easily modified for use in the company's other business divisions, and for use in "help desk" systems for companies throughout Northern Ireland.

3.2.4 Benefits of System

The benefits of the product support system are:

- *Ability to learn*. Case-based reasoning technology [4] is now a viable proposition. The system uses this technology to improve its capabilities as more discriminating cases are added.
- *Integration of case-based reasoning with conventional information system*. The case-based reasoning component of the system is intimately inter-linked with office automation software and a relational database to provide full information flow through the system.
- *Ease of use*. The system is designed to replace a current paper-based system and greatly extend the existing system's functionality. A forms metaphor is employed as users interact with the system and communicate with other users to sign off work.

- *Incorporation of product support manuals*. To provide support engineers with full background information, the system will soon incorporate all product support manuals, with accompanying graphical and pictorial information.
- *Full reporting facilities*. As all faults are time-stamped at each stage of their journey through the system, the reporting component of the system monitors their progress, and automatically generates digital memoranda reminding support staff of outstanding work.
- *Statistical analysis*. The statistical analysis component determines the efficacy of the product support, and give real-time information on current faults and the work load on the team's engineers.
- *Management information function*. Historical information may be automatically garnered for management information purposes, to show fault trends on product lines, and for use by the product's design team.

4. Conclusions

As knowledge based systems continue to be integrated with and embedded within conventional information systems, the need for a stand-alone methodology for the development of the KBS component may be called into question. Also, object-oriented modelling methods provide benefits to IS/KBS development such as intercalation between iterative development and prototyping, and the potential of reuse for major components of the system like static domain knowledge classes and user interface classes.

Although this paper advocates the mixing of various methods such as object-oriented modelling, knowledge elicitation, and prototyping, it should be emphasised that the result is not a rigorous methodology, but a collection of tools that help manage the comlexities, and avoid the potential pitfalls of joint IS/KBS development.

References

1. **Schreiber, A. T., Wielinga, B. J. And Breuker, J. A.** (Eds*) KADS: A Principled Approach to Knowledge-Based System Development*. Academic Press. London. 1993.
2. **Cosytec**. *CHIP V4 Manuals*. France. 1993.
3. **Van Hentenryck, P**. *Constraint Satisfaction in Logic Programming*. MIT Press. 1989.
4. **Schank, R. C. And Slade, S. B**. The Future of AI: Learning from Experience. *Applied Artificial Intelligence*. Vol. 5 pp. 97-107. 1991.