Hardy: Recent Developments

Julian Smart

1 Introduction

Hardy is AIAI's diagramming and modelling tool, which permits the creation of diagrams in a chosen format. In software analysis and design circles, it would be known as a meta-CASE tool; however, Hardy contains hypertext components which are not normally found in such systems. It also incorporates NASA's CLIPS expert system development tool; Hardy/CLIPS provides all the functionality of CLIPS, plus a suite of functions for interfacing between CLIPS and Hardy, thus allowing a high degree of customisation. This article reports technical progress on Hardy, as a prelude to a series of articles about how Hardy has been used to support AIAI projects.

Figure 1 shows a screen dump of a recent release of Hardy with a variety of tools open. These comprise the enhanced diagram type manager, a library of symbols, the Hardy/CLIPS development window, and a diagram.

2 The Gryphon project

Since we last described Hardy in *airing*, the tool has been through a hefty remodelling process which has resulted in many improvements. Hitachi Europe Limited (HEL) funded this work in a 12-month period from 1993 to 1994 on a project called ‘Gryphon’ by AIAI, and known as ‘Valise’ (lightweight CASE) within HEL. The principal goal of this project was to enhance Hardy to the point that all, or nearly all, graphical object-oriented analysis and design methods could be supported by Hardy. We left the actual diagram type creation and custom code to HEL, supporting them with a proprietary

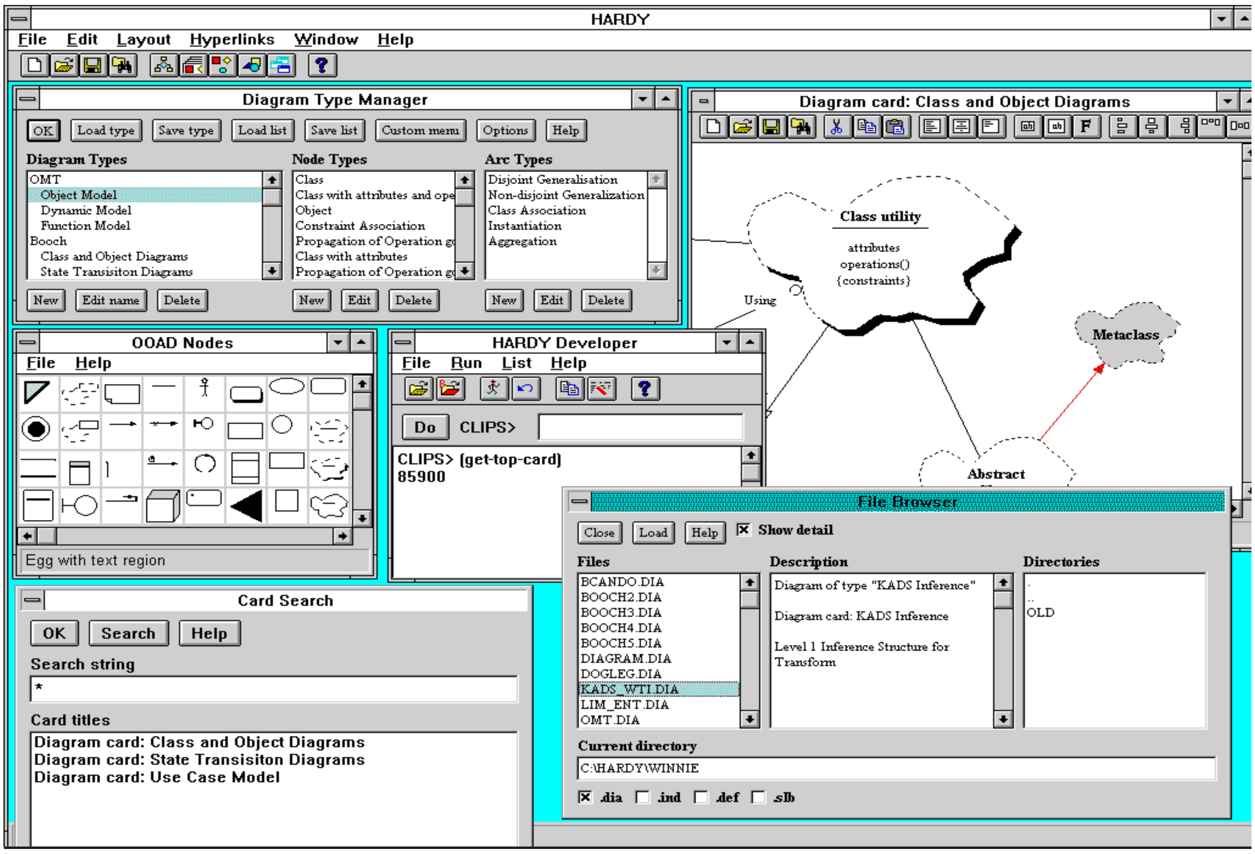


Diagram Types
OMT
 Object Model
 Dynamic Model
 Function Model
Booch
 Class and Object Diagrams
 State Transisiton Diagrams

Node Types
Class
Class with attributes and ope
Object
Constraint Association
Propagation of Operation g
Class with attributes
Propagation of Operation g

Arc Types
Disjoint Generalisation
Non-disjoint Generalization
Class Association
Instantiation
Aggregation

Egg with text region

```
CLIPS> (get-top-card)
85900
```

Search string
*

Card titles
Diagram card: Class and Object Diagrams
Diagram card: State Transisiton Diagrams
Diagram card: Use Case Model

[✓] Show detail

Files	Description	Directories
BCANDO.DIA	Diagram of type "KADS Inference"	.
BOOCH2.DIA		..
BOOCH3.DIA	Diagram card: KADS Inference	OLD
BOOCH4.DIA		
BOOCH5.DIA	Level 1 Inference Structure for Transform	
DIAGRAM.DIA		
DOGLEG.DIA		
KADS_WTI.DIA		
LIM_ENT.DIA		
OMT.DIA		

Current directory
C:\HARDY\WINNIE

[✓ .dia] .ind .def .slb

Figure 1: Hardy 1.36 running under Windows 3.1

translator (CLIPS to C) for delivery of the finished application without requiring CLIPS.

These are some of the major features added to Hardy:

- node/arc palettes for diagram cards, providing point and click image creation;
- addition of diagram card and main window toolbars, allowing common operations to be executed using point-and-click;
- a symbol library and node/arc symbol editors;
- editable symbol attachment points, to specify where arcs meet nodes;
- node images and arrowheads can be imported from metafiles;
- composite node images;
- the concept of node and arc annotations, with specifiable drop sites;
- containment: diagram fragments may be dropped into container node images;
- one-to-many multiway arcs, and self-linking arcs;
- a maximum of three, moveable arc labels on each arc;
- proper copy, cut and paste operations, with MS Windows clipboard support;
- a more consistent interface and links to on-line help in all dialog boxes;
- improved tree-drawing algorithm for supporting hierarchical diagrams;
- a convenient file browser supporting most Hardy file types;
- an alternative C++ - based programming interface;
- migration of the UNIX version from OpenLook to Motif;
- many more Hardy/CLIPS functions to support the new features and to help customisation.

The result of these enhancements is a system that HEL can now use as a very flexible CASE tool, encompassing the many kinds of diagrams that are found in the Booch, OOSE, Coad and Yourdon, and OMT methods. AIAI projects have benefited too, since the enhancements have been of a general nature. Features such as multiway arcs and composites are doing service in applications which perform process modelling and critical path analysis.

Figure 2 is an example of the kind of diagram that Hardy now supports: this is an OMT dynamic model, exhibiting containment, a self link, and multiway arcs. Other articles in this issue will provide more examples.

3 Public release

An exciting development for AIAI is Hardy's release for research and personal use. Although publicity has been minimal, interest has already been shown by researchers from surprisingly diverse backgrounds. Proposed applications include Petri net modelling in France, mind maps at Sharp's Oxford laboratories, executive information systems in Glasgow, knowledge modelling in Helsinki, KADS modelling in Montreal, manufacturing decision support in Galway, flowcharting in Georgia (USA), data acquisition at Oxford University, an ESPRIT project in the Aegean, CASE tool prototyping in Sunderland, and - my favourite - man-machine interfacing for nuclear power stations in Mexico!

We expect to receive useful feedback and examples from these projects. Hardy is distributed mainly through our World Wide

Web (WWW) pages and our file transfer protocol (ftp) server:

WWW:	*http://www.aiai.ed.ac.uk/~hardy/hardy.html*
ftp:	*ftp.aiai.ed.ac.uk/pub/packages/hardy/distrib*

The Sun Motif and Windows versions of Hardy are available as a limited demonstration for general distribution, which may be unlocked by agreeing to the terms of the personal licence, and filling out a WWW form. If your details are accepted, you will immediately receive a serial number which will unlock the demo copy.

4 Further development

As AIAI and its clients work with Hardy, we take on board suggestions and steadily improve the software. Much of Hardy's cross-platform strength depends upon another AIAI product, the multiplatform C++ class library wxWindows, which is being developed as part of the Hardy effort. The growing wxWindows user community (mostly on the Internet) is developing plans for extending the range of platforms and functionality covered, so this will be good news for Hardy. We can expect to see it on platforms such as the Mac, and possibly NeXTStep, in the medium term; and tricky areas such as OLE-2 and database support are being considered in a multiplatform context.

But of course, the real future of Hardy lies in the success of its applications. We hope Hardy will continue to enable our clients to receive the modelling functionality they require, at minimum risk and cost, and with maximum flexibility.

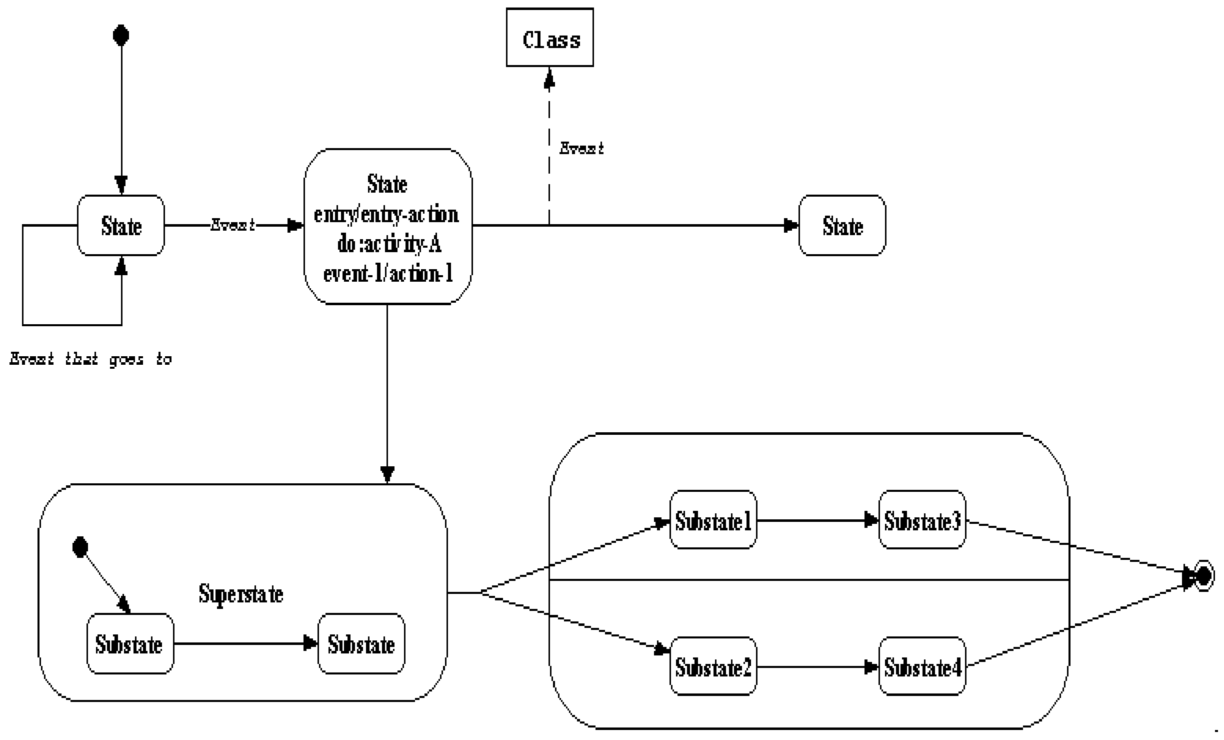


Figure 2: OMT Dynamic Model

TOPKAT — Supporting Knowledge Acquisition and the CommonKADS Methodology

John Kingston

1 Introduction

TOPKAT (The Open Practical Knowledge Acquisition Toolkit) is a hypertext and diagram-based toolkit which supports various knowledge acquisition techniques, as well as supporting much of the CommonKADS modelling framework.

TOPKAT has been implemented using Hardy and Hardy/CLIPS. The facility in Hardy to define a number of different diagram types allowed the production of modelling tools for a wide range of graphical formalisms with little effort; Hardy/CLIPS was then used to automate (wholly or partially) many common operations.

TOPKAT consists of a hierarchy of hypercards. The card at the top of the hierarchy (shown in the top left hand corner of Figure 1) acts as an index for the different facilities available. Diagrams are drawn on newly created hypercards which are expansions of a particular pre-defined card, and which therefore share the same diagramming type.

TOPKAT currently supports the following knowledge acquisition techniques:

- transcript analysis;
- laddered grid;
- card sort;
- repertory grid.

TOPKAT also provides support for representing the following elements of the CommonKADS Expertise Model:

Domain Knowledge:

- Domain ontology;
- Domain models;
- Model ontology;
- Model schema.

Inference Knowledge:

- Inference structures;
- Library of inference structures.

Task Knowledge:

- Task structures.

In addition, facilities exist within TOPKAT for representing parts of the CommonKADS Task Model, Communication Model and Design Model.

The transfer of knowledge between the knowledge acquisition techniques and the CommonKADS representations proved to be a piece of work which was of considerable theoretical and practical interest. The techniques used are described in detail in [1]; as an example, the facilities for automated transcript analysis are outlined here.

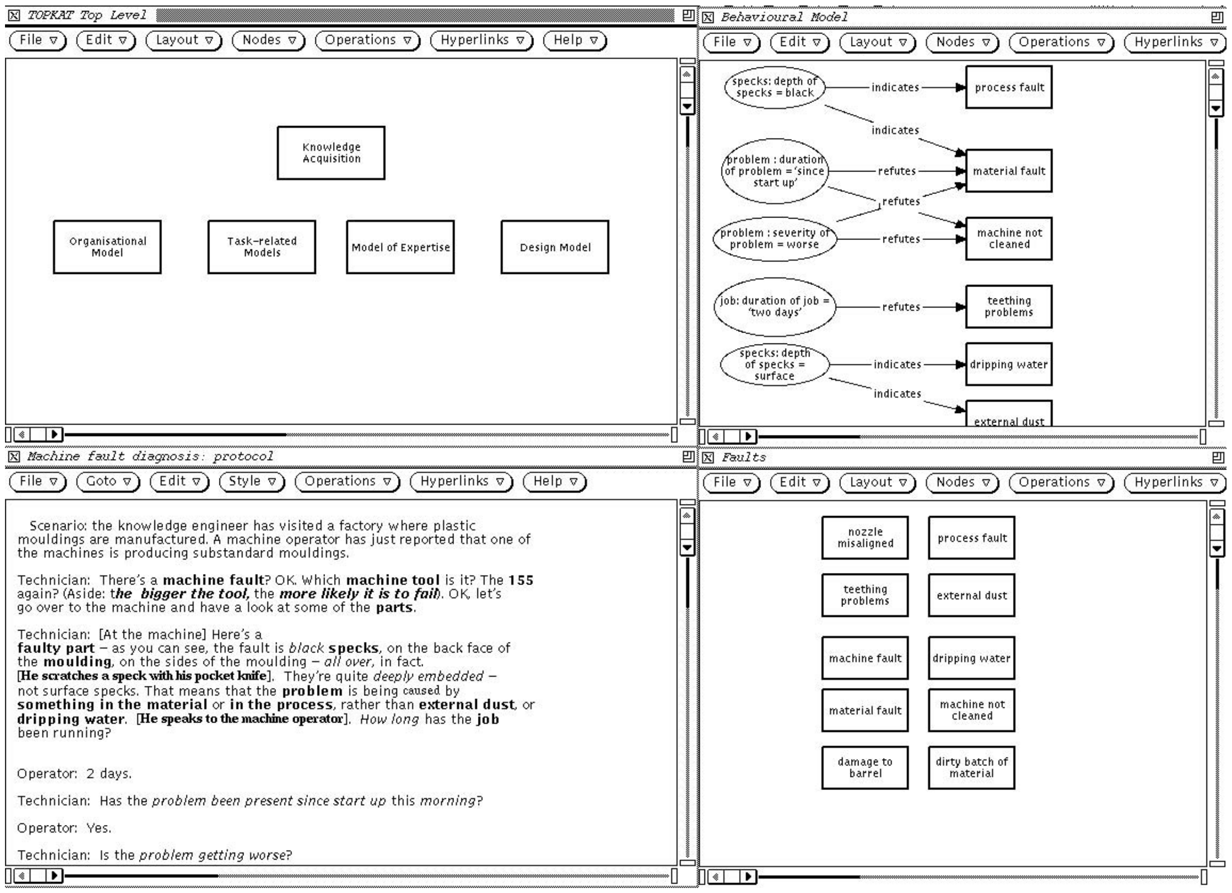
TOPKAT has been designed so that a file which is displayed in a hypertext card can be exported to a lexical tagging package, which identifies the word class (e.g. noun, adjective, verb) of each word. This ‘tagged’ file is then re-imported into TOPKAT, and the tagged file is used to make a provisional hypertext mark-up of the file, according to word class. Once this has been done, TOPKAT identifies all the nouns in a transcript, lists them, and then sorts them in inverse order of word popularity, i.e. words which are used least frequently in normal English are placed at the top of the list. This is achieved by interfacing with a dictionary package which provides measures of word frequency. The user is then presented with this list, and asked which of the nouns on the list represent important concepts in the domain.

Once this has been done, a simple procedure identifies adjectives which are attached to concepts, and asks if these represent properties, or values of properties, of that concept. It is hoped that future developments in TOPKAT will include integration with a parsing package rather than just a tagging package, allowing fuller identification of adjectives and also identification of verbs which link concepts, and may therefore represent relations.

TOPKAT is currently being re-implemented in version 6.0 of CLIPS, which permits full integration of object hierarchies with CLIPS' other facilities. This feature is being used to allow CLIPS objects to serve as a knowledge repository, with Hardy being used as a tool for visualising and manipulating that knowledge; this is achieved using a small set of event handlers (daemons) which create Hardy nodes or arcs to represent a set of CLIPS objects, and another set of handlers which generate CLIPS objects whenever nodes or arcs are created in Hardy. This feature will allow the functions within TOPKAT which perform verification, analysis and automated linking to be implemented entirely in CLIPS, thus increasing the portability of TOPKAT.

2 References

[1] Kingston, J.K.C, 'Linking Knowledge Acquisition with CommonKADS Knowledge Representation', Proceedings of BCS SGES Expert Systems'94 conference, Cambridge, December 1994. Also available as AIAI-TR-156



Scenario: the knowledge engineer has visited a factory where plastic mouldings are manufactured. A machine operator has just reported that one of the machines is producing substandard mouldings.

Technician: There's a **machine fault**? OK. Which **machine tool** is it? The **155** again? (Aside: t***he bigger the tool,*** the ***more likely it is to fail***). OK, let's go over to the machine and have a look at some of the **parts**.

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

Operator: 2 days.

Technician: Has the *problem been present since start up* this *morning*?

Operator: Yes.

Technician: Is the *problem getting worse*?

Figure 1: Screendump showing some of TOPKAT's hypercards

Developing Belief Networks using Hardy

Ian Harrison

1 Introduction

At AIAI we have used belief networks in client projects as a method for reasoning under uncertainty. Hardy has been used on these projects to draw the belief network diagrams which essentially capture a model of expertise about a domain. In addition, a graphical rule editor has been developed using Hardy/CLIPS. Once a diagram has been drawn and rules defined it is possible to run the model (the diagram).

This paper first introduces belief networks and then discusses their use in an industrially funded client project called SPIRIT. After that, it is shown how Hardy was used to support the development of the belief networks within the SPIRIT project, and finally conclusions are given.

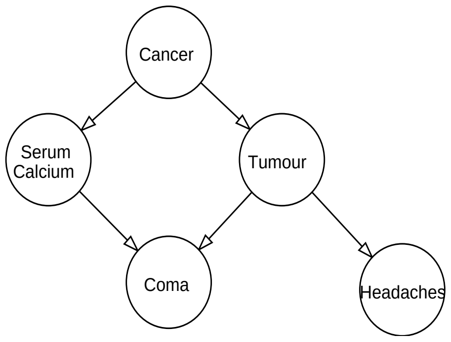


Figure 1: Example belief network

2 Belief Networks

Belief networks are basically directed acyclic graphs (see Figure 1) and they are becoming an increasingly popular knowledge representation for uncertain reasoning [7]. Amongst other names given to belief networks are Bayesian (belief) networks, knowledge maps, probabilistic causal networks and qualitative probabilistic networks. The nodes in a belief network represent a random variable, or uncertain quantity, that can take two or more possible values. The arcs signify the existence of direct influences between the linked variables, and the strength of these influences are quantified by forward conditional probabilities. Belief networks are a way of modeling knowledge about a domain that contains uncertainty.

Within a belief network the basic computation is to calculate the belief of each node (the node's conditional probability) based on the evidence that has been observed. Various methods have been developed for evaluating node beliefs and for performing probabilistic inference. The most popular methods are those of Pearl [6] and Lauritzen and Spiegelhalter [4]. Similar techniques have been developed for constraint networks in the Dempster-Shafer formalism [8]. In addition to numerical representations of uncertainty other work has concentrated on non-numerical uncertainty handling, e.g. [1,2,5]. However, all these schemes are basically the same - they provide a mechanism to propagate uncertainty in the belief network, and a formalism to combine evidence to determine the belief in a node.

3 Use of belief networks in the SPIRIT project

The SPIRIT project [3], which was funded by the oil industry, aimed to develop a prototype of the next generation of software for the task of well test interpretation. The aim was to use knowledge-based techniques to provide a decision support capability for petroleum engineers who have to interpret oil well pressure test data. An important aspect of SPIRIT was the requirement for managing uncertainty. From speaking to petroleum engineers it became clear that much of the uncertainty in data was expressed in non-numerical terms. When speaking to experts in well test interpretation it was clear that they used symbolic terms such as "very likely" or "strongly supported" to describe how likely a conclusion was based on a given set of data.

Based on this, it was decided that a belief network approach would be suitable for the SPIRIT project. Three types of nodes were identified as being useful for knowledge acquisition purposes, but all of these had the same underlying behaviour; that is they were nodes whose level of belief was determined by the combining function attached to that node. Figure 2 shows a very small part of the belief network in SPIRIT. At the bottom are evidence nodes (ovals) which represent the mapping of raw data or quantitative interpreted data into qualitative interpreted data. Cluster nodes (rectangles) are engineering or geologically significant groupings of evidence. Cluster nodes can, in turn, support or weaken model nodes (diamonds) which represent the goal of the belief network.

4 Using Hardy to develop belief networks

Using Hardy it was possible to devise a diagram type for belief networks (see Figure 2). For the SPIRIT project this enabled a knowledge engineer to work with an expert creating, modifying and updating diagrams which represented a model of his expertise.

Using Hardy/CLIPS, it was possible to develop a graphical rule editor which allows the definition of combining rules for each

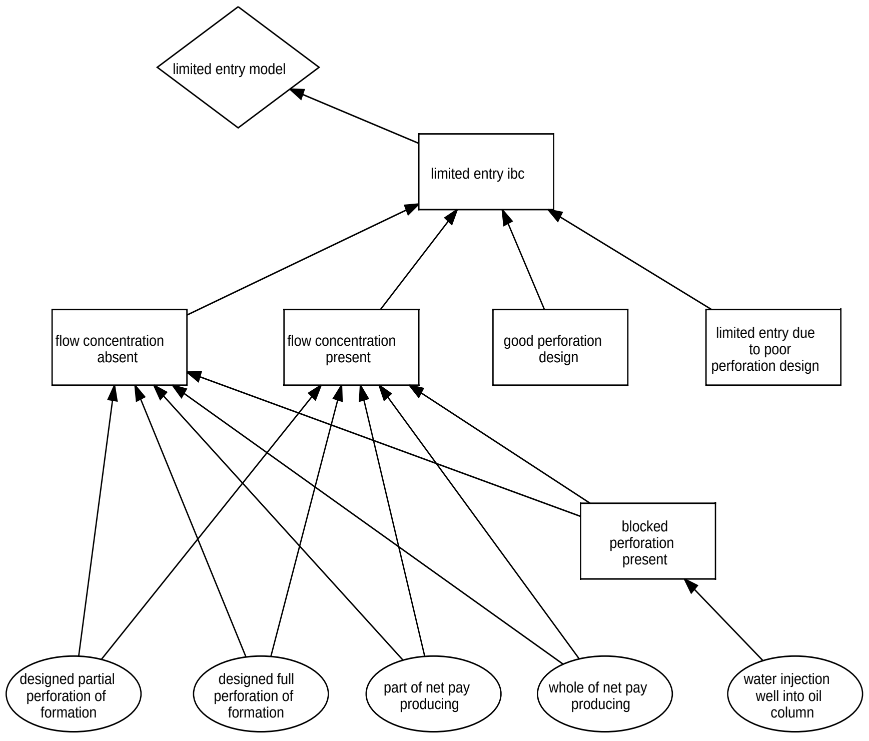

limited entry model
limited entry ibc
flow concentration absent
flow concentration present
good perforation design
limited entry due to poor perforation design
blocked perforation present
designed partial perforation of formation
designed full perforation of formation
part of net pay producing
whole of net pay producing
water injection well into oil column

Figure 2: Part of the belief network in SPIRIT

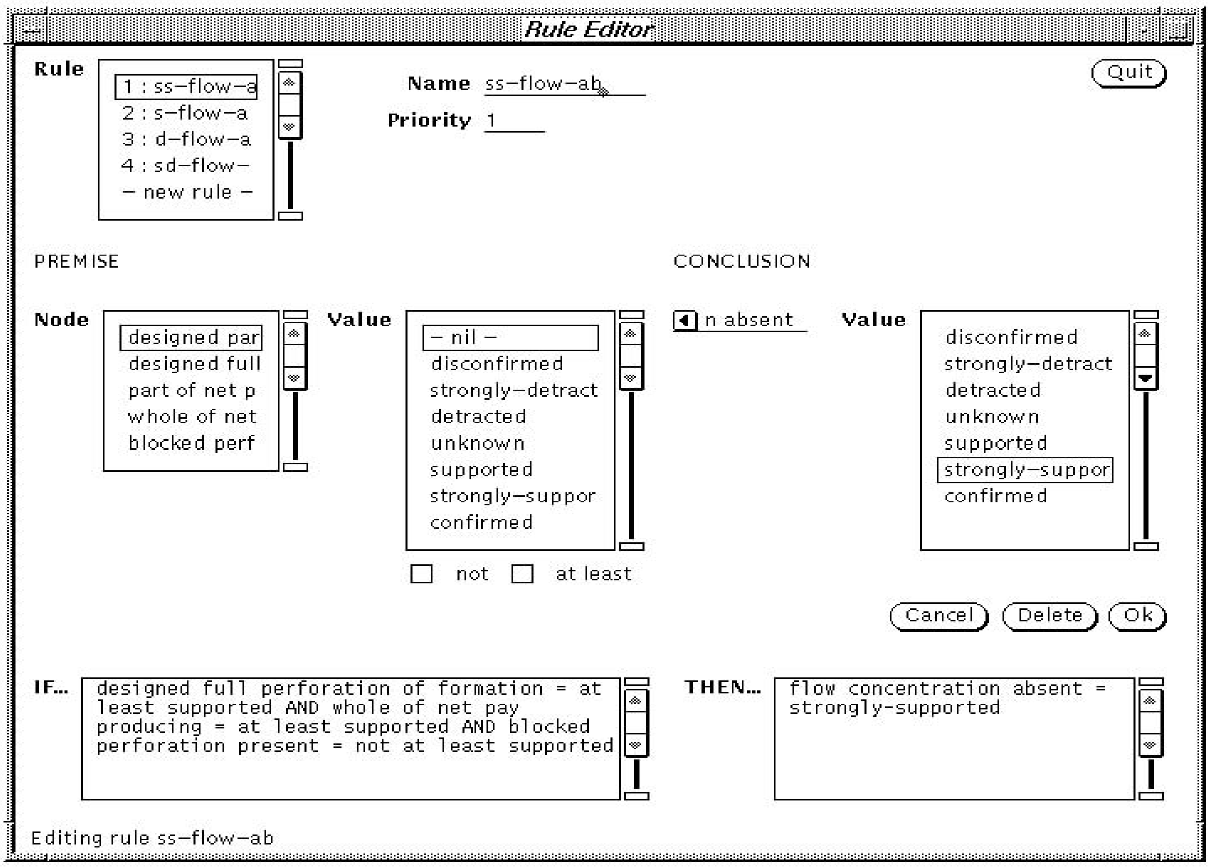
node within the network (see Figure 3). The combining rules determine the level of belief in a node given the level of belief in its connected nodes. For the SPIRIT project the representation of uncertainty used qualitative values instead of probabilities. In this case belief in a proposition was expressed as one of seven values -

confirmed, strongly supported, supported, unknown, detracted, strongly detracted or rejected.

With the belief network diagram drawn and combining functions defined for the network, it is then possible to actually run the network. The user can set the levels of belief of root nodes in the network and these are propagated through the network according to the defined combining rules.

5 Conclusions

Belief networks are an intuitive way of representing and reasoning with uncertainty, whether probabilistic or qualitative. On the SPIRIT project Hardy proved to be an excellent tool for knowledge acquisition. Hardy enabled a model of expertise to be captured in a graphical manner such that it was understandable to the expert, in a way which rules alones could not be. In addition, be developing the graphical rule editor, the combining rules for each node in the network could also be captured. The network could then be run, allowing the expert to see the effect of the rules; errors within rules could thus be easily detected.



Name ss–flow–ab
Priority 1

IF... designed full perforation of formation = at least supported AND whole of net pay producing = at least supported AND blocked perforation present = not at least supported

THEN... flow concentration absent = strongly-supported

Editing rule ss–flow–ab

Figure 3: Graphical rule editor for a qualitative belief network

6 Reference

[1] Z. An, D.A. Bell, and J.G. Hughes. Res: A relative method for evidential reasoning. In D. Dubois et al., editor, 8th Conference on Uncertainty in Artificial Intelligence, pages 1-8. Morgan Kaufmann, San Mateo, California, 1992.

[2] P.R. Cohen, D. Day, J.D. Lisio, M. Greenberg, R. Kjeldsen, D. Suthers, and P. Berman. Management of uncertainty in medicine. Int. J. of Approximate Reasoning, 1:103-116, 1987.

[3] J. Fraser, and I. Harrison. Modelling Expertise Using Belief Networks, in Bramer A and Macintosh A L (eds), Research and Development of Expert Systems X, Proceedings of Expert Systems 93, Robinson College, Cambridge, UK, December 1993. Also available as AIAI-TR-134, October 1993.

[4] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. In G. Shafer and J. Pearl, editors, Reading in Uncertain Reasoning, pages 415-448. Morgan Kaufmann, San Mateo, California, 1990.

[5] T.Y. Leong. Representation requirements for supporting decision model formulation. In B.D. D'Ambrosio et al., editor, 7th Conference on Uncertainty in Artificial Intelligence, pages 212-219. Morgan Kaufmann, San Mateo, California, 1991.

[6] J. Pearl. Fusion, propagation and structuring in belief networks. Artificial Intelligence, 29(3):241-288, 1986.

[7] R. D. Shachter, B. D'Ambrosio, and B. A. Del Favero. Symbolic probabilistic inference in belief networks. In AAAI-90, volume 1, pages 126-131, 1990

[8] G. Shafer. Perspectives on the theory and practice of belief functions. Int. J. of Approximate Reasoning, 4:323-362, 1990.