

Capturing and Using Knowledge

A White Paper by Attar Software

The screen shots in this paper are taken from the earlier XpertRule KBS (Knowledge Based System) although the same principles apply to XpertRule Knowledge Builder.

Organisations today are making every effort to improve their products and services while controlling costs. Many have undertaken Business Process Re-engineering (BPR) to find better ways to perform complex tasks, or to improve and automate their operations. Performing complex tasks requires the know-how of the organization's experts and specialists. Unfortunately, there is usually a shortage of such people and their knowledge is almost always locked away in their heads. If they leave the organization, they take their knowledge with them. The challenge is to capture and automate their knowledge to make it available to others. Accomplishing this type of Business Process Automation (BPA) is extremely difficult using conventional programming tools. Putting this knowledge within conventional programming code simply replaces one lock with a double lock - you may need an expert and a programmer to get at it! Since you cannot clone your experts, you must find another way.

Knowledge Acquisition

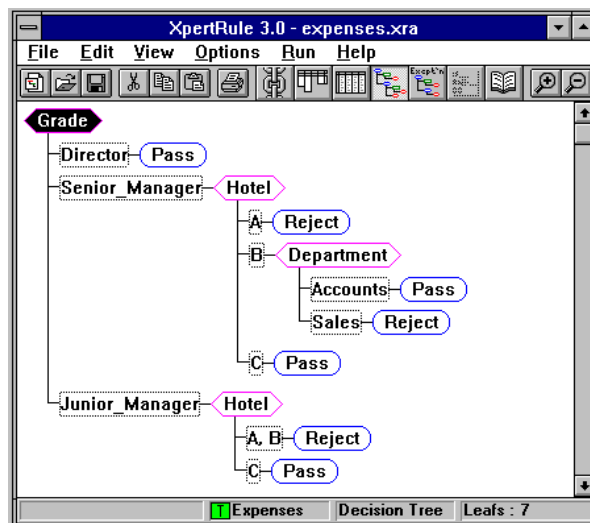
Knowledge acquisition is an iterative process, best performed with interactive tools engineered specifically to capture knowledge. Once captured and recorded, this knowledge base can then be transformed into automated systems to:

- Enhance human decision making by offering advice.
- Free experts from repetitive routine decisions for more productive and rewarding work.
- Ensure that decisions are made in a consistent way and as speedily as possible.
- Retain the organization's expertise in a readily maintainable form.

Eliminating the nightmare of programming and maintaining complex logic

As every developer knows, complex decision making logic can be a nightmare to specify, program and maintain. Capturing decision making knowledge is especially difficult because it is rarely, if ever, documented and must be elicited from experts or other users. Once specified, the knowledge is conventionally written in program code, which may consist of many IF..AND..OR..THEN..ELSE program statements. It is difficult to debug and test all possible decision paths, let alone optimize the flow to maximise performance. Such programs, over time, become impossible to maintain. Quality knowledge representation software could eliminate this programming nightmare by providing easy-to-use graphical facilities for capturing, structuring, representing and maintaining knowledge.

The building blocks of Knowledge - Decision Trees and Examples



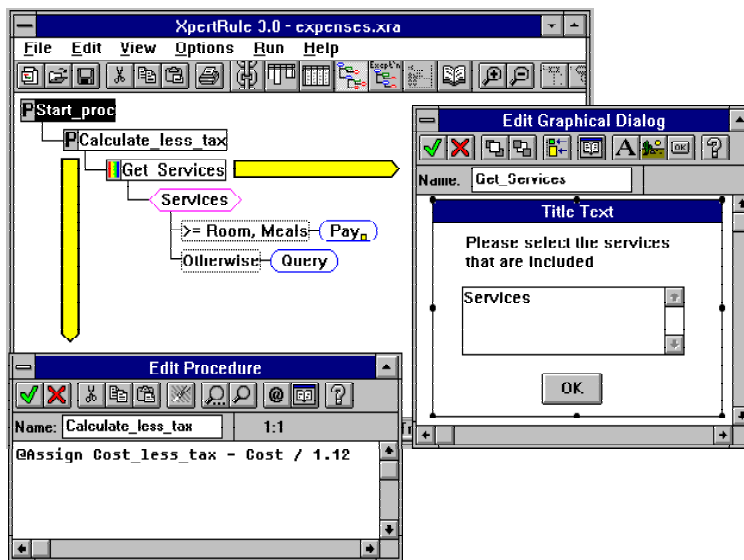
Decision Trees

Decision Trees are one main knowledge representation method. This representation is both compact and efficient, far better than other forms used by conventional rule based systems. See our on-line tutorial on Using Decision Trees for a detailed example.

The graphical nature makes them more understandable and the inference from trees can be orders of magnitude faster than inference from rules because of the elimination of the need to search rule bases. A tree can represent many 'rules' and when you execute the logic by following a path down it, you are effectively bypassing rules that are not relevant to the case in hand. You do not have to look at every rule to see if it 'fires' and you also take the shortest route to the correct outcome.

Executable process flow diagrams

Business analysts often draw process diagrams to show program logic. Decision trees enable you to show decision making logic in an easy-to-understand graphical form.



The trees can show both decision making and the structured flow of procedural processing which may for example include calculations, database access or graphical dialogs.

The screenshot shows the XpertRule - expenses.xra interface with a data table. The table has columns for Grade, Department, Hotel, Expenses, and Comment. The data is as follows:

	Grade	Department	Hotel	Expenses	Comment
1	Director	*	*	Pass	
2	Senior Manager	Accounts	A	Reject	
3	Senior Manager	Accounts	B	Pass	
4	Senior Manager	Accounts	C	Pass	
5	Senior Manager	Sales	A	Reject	
6	Senior Manager	Sales	B	Reject	
7	Senior Manager	Sales	C	Pass	
8	Junior Manager	Accounts	A	Reject	
9	Junior Manager	Accounts	B	Reject	
10	Junior Manager	Accounts	C	Pass	
11	Junior Manager	Sales	A	Reject	
12	Junior Manager	Sales	B	Reject	
13	Junior Manager	Sales	C	Pass	

At the bottom of the window, there are buttons for 'Expenses', 'Examples', and 'Number examples: 13'.

Examples (Cases)

Examples, or cases, are one of the easiest ways for humans to express knowledge. An expert, working alone or with a developer, simply specifies the possible results (outcomes), of a process and lists the factors (attributes) which affect these outcomes.

Specifying knowledge through examples, relating outcomes to attributes, exercises the experience of the expert.

The Need for Knowledge Distillation

While Examples are very natural ways for experts to express knowledge. They are not ideal representations for validating knowledge, because:

- They may contain inconsistencies, such as cases which contradict each other.
- They are not necessarily efficient. For example, some attributes may be redundant.
- They may not represent the knowledge completely. The knowledge could still contain gaps.

How do you know if rules conflict? How do you know if there are gaps in the knowledge? How do you know if certain attributes are redundant?

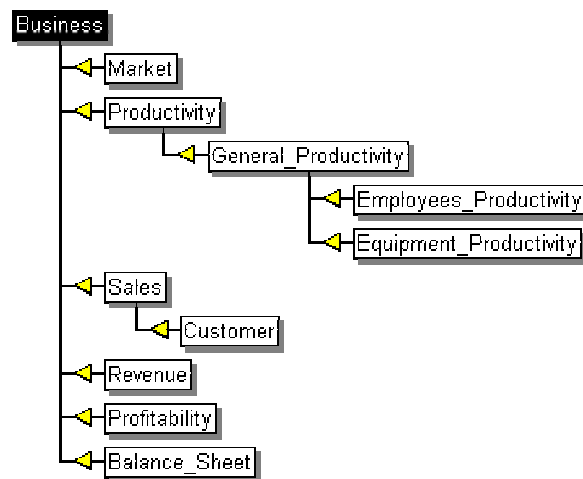
Rule Induction

From examples, XpertRule uses a process called Rule Induction to automatically generate a Decision Tree that will express the same logic in a more efficient way. This provides a solution to all the problems cited. Conflicts and gaps in the logic are made visible - and they also point out where the problem can be resolved in the original Examples. Alternatively, the Decision Tree itself can be easily edited.

The end result is a decision tree that is a form of knowledge representation that ensures:

- No gaps in the knowledge.
- Fast execution of the knowledge.
- An efficient priority order in which to seek attributes (factors). The rule induction process builds the tree by branching on the most 'information bearing' attributes in order of importance in deciding the outcome. This can be overridden by the developer, if required.

Rule induction is also used to derive knowledge from data. Attar's Miner discovers hidden patterns and relationships in data and is used extensively for Data Mining.



Structuring large applications

When building real life systems, all of the techniques of knowledge capture can be used as building blocks. The application can be broken down into components.

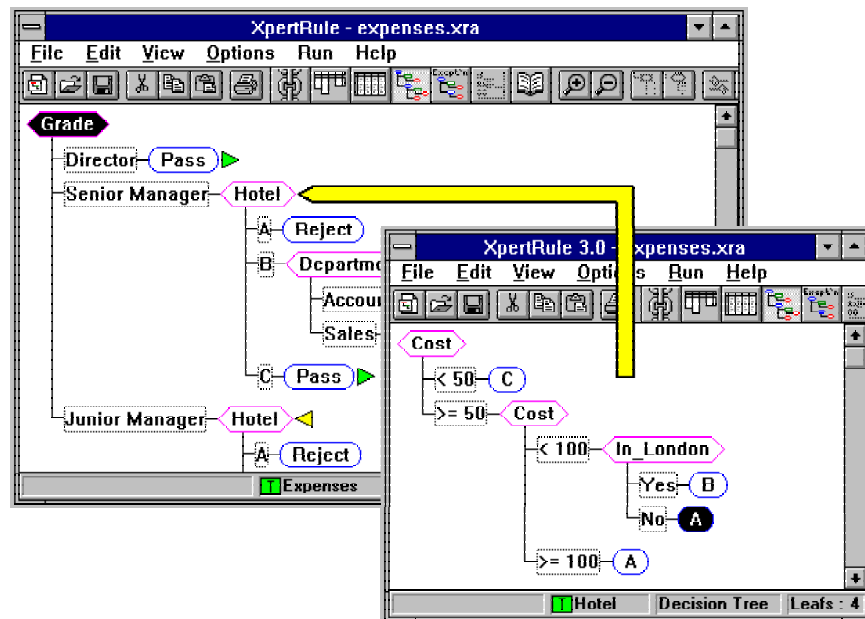
For example, in deciding how to assess the risk of lending to a small to medium sized company, a financial institution may look at several aspects - the market, sales, revenue, balance sheet etc. Each one of these may use a different knowledge capture technique. Sales using a decision tree, revenue an exception tree, the market - a set of examples. Such building blocks are units of knowledge, or decision making 'tasks', which, while the overall complexity of the system is high, make the components understandable both as individual units, and as a hierarchy of knowledge tasks.

The task Map enables the developer to review and navigate around their application, and provides a very illustrative overview of the application for ongoing maintenance. You can simply zoom in and out of the tasks.

Executing the Knowledge

Unlike process charts, Decision trees can be tested immediately. The Inference Engine can be used to 'run' the application at any time. Automatic default user dialogs are generated, enabling the testing of prototypes with no developer effort. Nesting of trees can be used to relate sub-tasks to a main task.

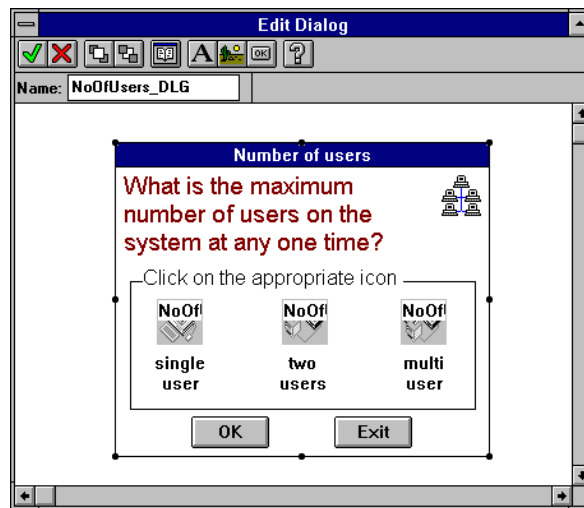
In this example, when the inference reaches the HOTEL attribute, a sub-task is executed to evaluate the desired outcome. The sub-task itself contains a decision tree. This processing is transparent to the end user.



Integrating Applications

To deliver complete solutions, the knowledge must be able to be integrated with other applications as well as the user interface being customised.

- An extensive command language to manipulate attributes, strings, arrays, perform calculations, access data files and integrate with external programs.
- Connectivity to hypertext help facilities.
- ODBC and DLL support to link your applications to other programs and data sources, plus COM+ automation.



The graphical dialog editor can be used to create custom GUI screens that use text, graphics and the typical Windows controls that users expect - Buttons, Scroll bars, 'Hot spots', List boxes, Fonts, Colours and Graphics, etc.

Reports can use HTML (Hypertext markup language) and be displayed with inbuilt browser functionality or via calling an external browser (both versions). This enables Intranet/Internet pages to be called up, or even created on-the-fly by to include variables and graphics generated at run time.



Using industry standard middleware, such as Open DataBase Connectivity (ODBC), enables you to access database servers through embedded SQL within your applications. Using Client-Server computing enables you to make the best use of processing power by offloading or distributing the workload across networks to dedicated servers hosting the data.