THE MAKE PROJECT: MAINTENANCE TOOLS FOR KNOWLEDGE BASED SYSTEMS

Trevor Bench-Capon and Frans Coenen

Department of Computer Science
University of Liverpool
PO Box 147
Liverpool
England

phone: 051-794-3698
e-mail: frans@uk.ac.liv.cs

One of the major obstacles to the routine exploitation of Knowledge Based and Expert Systems, is the difficulty of validating the knowledge base, and of maintaining it in a state which reflects current knowledge. This is of particular importance for systems based on law or regulations, where it is vital that the knowledge base be a true reflection of the legal position, and where there is a constant stream of changes to the correct legal position.

Maintenance Assistance for Knowledge Engineers (MAKE) is a project designed to explore these issues, and to build a set of tools which will support the validation and maintenance of knowledge bases deriving from regulations. These tools include facilities to examine the structural features of the knowledge base, so as to guard against redundancy, non-provability and contradiction; facilities to identify parts of the knowledge base jeopardised by changes in the domain, or in the understanding of the domain; and facilities to perform a variety of "house keeping" tasks. The paper firstly analyses the different types of change that may be required to maintain the knowledge base, and then proceeds to describe the set of tools developed in the MAKE project to accommodate these changes.

THE MAKE PROJECT: MAINTENANCE TOOLS FOR KNOWLEDGE BASED SYSTEMS

Trevor Bench-Capon and Frans Coenen

1. INTRODUCTION.

Knowledge Based Systems (KBSs) have been in existence for several decades now, but their general acceptance is still limited. It is suggested that one principal reason for this is the difficulty in maintaining such systems due to the nature of the expert knowledge used [BEN91a]. This tends to be dynamic, in that the knowledge will change as new discoveries are made and expert opinions alter over a period of time. Thus KBSs, if they are not maintained, quickly become obsolete or inaccurate. This is even more the case in legal domains where the knowledge required may change overnight if legislation is altered.

The maintenance of KBSs thus tends to be adaptive maintenance rather than the corrective or perfective maintenance associated with conventional systems [COE90]. Further the structure of KBSs is such that traditional maintenance techniques are not applicable. Given the above a surprisingly small amount of research work has been carried out to investigate the maintenance of KBSs. What work has been carried out has largely been concerned with formal consistency checking of Rule Bases (RBs) such as in the COVADIS system [ROU88], the debugging of RBs such as provided by the TEIRESIAS system [DAV84], or the syntactical editing of RBs within KBS development toolkits.

In this paper, a number of tools to assist the KBS maintenance engineer are outlined. The tools form a suite of tools under development as part of the Maintenance Assistance for Knowledge Engineers (MAKE) Project further details of which can be found in [BEN91a]. The MAKE project is a two year collaborative project between Liverpool University, ICL and British Coal. The tools are essentially designed to be used in conjunction with KBSs built using the MADE (MAKE Authoring and Development Methodology)development environment and methodology.

Knowledge analysis using MADE results in a Class Hierarchy (CH), giving a vocabulary for the domain, and a Rule Base (RB) represented in an intermediate representation called MIR (Make Intermediate Representation). This is described in [BEN91b]. The resulting knowledge about the domain can be considered to exist at three different levels. At the top most level are the source documents themselves, at the second level is in the MIR and at the third the fine grain executable representation produced when the MIR is compiled into CMIR (Compiled MIR).

2. OVERVIEW OF THE MAKE AUTHORING AND DEVELOPMENT ENVIRONMENT (MADE).

MADE is a KBS development environment based on KANT (Knowledge ANalysis Tool). This is a hypertext like knowledge analysis tool originally built to assist in the development of a KBS to provide decision support for Department of Social Security (DSS) Adjudication Officers in the assessment of claims for benefits in local DSS offices [STO89]. This system was called the Local Office Demonstrator (LOD) System and was one of three applications built as part of the Alvey DHSS Large Demonstrator Project aimed at demonstrating the viability of KBS decision support in large, legislation based organisations [FOR91]. It is ideally suited to the construction of KBSs in domains, such as legal domains, where the source knowledge is comprised of a significant amount of textual material, by assisting the knowledge engineer in the analysis of these source documents.

The design of MADE revolves around three base windows, the KANT, MADE and MAPPE (Make APPlication Environment) Windows. From the KANT base window sources can be selected and analysed. This is where the Knowledge Analysis of the Application is carried out. The Result is a Rule Base and Class Hierarchy in MIR. The MIR Rule Base and Class Hierarchy are then compiled into CMIR, the target representation used in the MAKE Inference Engine (MIE). Other target representations could be used equally well, for example Kappa or NExpert.

3. OVERVIEW OF MIR.

MIR is essentially a simple language to define Objects and rules using first order predicate logic with some extensions, for example to handle arithmetic.

The CH in the MIR consists of a top level Class with Sub-Classes branching of it describing different object types. Each Class has a unique Class name describing an object and a number of Slots describing Attributes of that object, and the possible Values that the Attribute may have. A feature of the CH is that Sub-Classes inherit Attributes of Super-Classes. Thus a Sub-Class is always a strict specialisation of its Super-Class.

The Rule Base will consist of a set of Rules typically all branching out from a single top level node. Rules are task dependent, and are thus arranged in a hierarchy Headed by a top level Rule representing the goal to be attained: further Rules then branch of this top level Rule down to a number of leaf Rules. Predicates found in Rules are therefore either "leaves", in which case their truth will be ascertained by direct reference to the CH or the user, "roots", where they are what the system is intended to establish, or "intermediate", where they need to be established from the KB in order to establish some root.

4. KBS MAINTENANCE.

It is not the intention of the MAKE project to address major maintenance tasks which may necessitate the entire rebuilding of the system. The aim is to address minor adaptive maintenance only, i.e. maintenance resulting from the day to day changes in the source material due to changes in legal texts, the application and operation of the law etc. In this context the maintenance required can be considered under a number of Headings a) RB maintenance, b) CH maintenance, c) changes to the source data and d) validation.

4.1. RB Maintenance.

The maintenance of the RB will involve one or more of the following activities:-

M1 The introduction of a new Rule.

M2 The modification of an existing Rule.

M3 The removal of an existing Rule.

The effect of introducing a new Rule may be unwanted redundancy or subsumption, or the creation of a missing branch, a hard contradiction or soft inconsistency.

The introduction of a new Rule will thus involve checking for the following:-

C1 Redundancy or subsumption.

C2 Missing Rules or branches.

C3 Hard inconsistency.

C4 Soft inconsistency.

The removal of a Rule may also result in the creation of a missing branch or cause a section of the RB to become redundant. Therefore when removing a Rule checks C1 and C2 should be implemented. The modification of a Rule has the same effect as removing a Rule and introducing another. Hence the methods outlined above for the introduction and removal of Rules can be used in sequence.

It would be rare for a maintenance session to consist of only the removal or introduction of a single Rule. In most cases a maintenance session will involve all three types of RB maintenance, i.e. M1, M2 and M3. It is therefore proposed that on completion of any KB maintenance checks for C1 to C4 should always be carried out. Two RB maintenance tools are therefore proposed:-

T1 The Rulemap.

T2 Hard Contradiction and Soft Inconsistency Identification.

4.2. CH Maintenance.

A Maintenance session involving the CH may include one or more of the following:-

M4 The modification of an existing Slot by introducing a new Value.

M5 The modification of an existing Slot by removing an existing Value.

M6 The modification of an existing Class by introducing a new Slot.

M7 The modification of an existing Class by removing an existing Slot.

M8 The introduction of an entire new Class.

M9 The removal of an existing Class.

4.2.1. Introducing or Removing a Value.

One of the most basic actions in the maintenance of the CH is the addition of a possible Value to a Slot. In practice a Value will be added to a Slot as a consequence of the introduction or modification of a Rule. The allocation of this additional Value to an existing Slot will not generally effect the operation of any established Rules or the existing CH. The exception to this is if Rules exist that use the possible Values of a Slot to express negation. Thus the Rules that contain the Attribute to which a Value is to be added need to be identifyed so that the effect of introducing this Value can be determined. For this purpose it will, in some cases, be necessary to go down to the fine grain level of representation.

Removing a Value from a Slot will jeopardise all Rules which make use of that Value either in the Head or the Body of the Rule. These Rules must therefore be identified and presented to the maintenance engineer so that a decision can be made on whether it is appropriate to remove the Rule, remove the Atom containing the removed Value, or modify the Rule. A tool to allow the identification of jeopardised Rules as a result of removing and introducing Values to and from Slots in the CH is there fore required. A more general tool to identify jeopardised Slots and Rules as a result of changes to the Source data or changes to the Rule Base or CH would however be more useful. Thus:-.

T3 Jeopardy Tool.

Because the inheritance mechanism used, the CH insists on strict specialisation, a Value can only be added to an Attribute at the highest level at which that Attribute appears. If the Rule which motivates the introduction of the new Value refers to a Class which inherits the Attribute from a Super-Class, either the Value must be added to the Super-Class, or some new Attribute must be created in the Class in question. If the Value is added to the Super-Class, of course, the Rules for that Class in which the Attribute appears are jeopardised. Therefore, before a new Value can be added, the user should be confronted with the Class which introduces the Attribute to the CH, which may not be the Class mentioned in the Rule which motivated the introduction of the Value, and the Value added to this Class. If the Class to which the Attribute is added is not a leaf Class, this process would be facilitated by a tool which walks down the CH so that the user is able to determine the correct point, on each path, at which the new Value should cease to apply.

Note that the removal of a Value will also cause that Value to be removed from all Sub-Classes of the Class from which it is removed. It will therefore be necessary to walk down the hierarchy repeating the process for all these Sub-Classes, until the Value has been "specialised out".

A CH navigation tool therefore suggests itself:-

T4 CH Navigation Tool.

4.2.2. Introducing or Removing an Attribute.

An Attribute can be added to a Class in two ways. Either it can be added directly to the Class, or it can be added to a Super-Class, and so added to the Class in question indirectly by inheritance. Thus if a Rule needs to mention a new Attribute for some Class, the first step should be to walk up the CH to determine the appropriate point at which the Attribute should be introduced into the hierarchy. Note, however, that adding it to a Super-Class will cause all the Sub-Classes of that Class to take on the Attribute, not only those on the path walked up. Once introduced into the hierarchy Rules may be written using the

Attribute. These can then be subject to the usual checks for new Rules already described. The final stage will then be to walk down the CH specialising the Values of the Attribute as appropriate, until a point range is reached on every downwards path. The process of adding a new Attribute to an existing Class will thus also involve the use of the CH Navigation Tool (T4).

The process of removing an Attribute from an existing Class can be regard as removing a set of Values. Inheritance, however, necessitates that the Attribute must also be removed from all the Super-Classes up to the point in the hierarchy at which it was introduced. The best approach would therefore be to commence the removal at this point, and then to walk down the CH to determine at which point (on each relevant path) at which the Attribute should now be introduced to the hierarchy. Class(es) at which it is now introduced and their Sub-Classes will not be affected. Thus CH Navigation Tool will also be of relevance here.

4.2.3. Introducing or Removing a Class.

The point in the hierarchy in which the Class should be introduced will be best determined by the Attributes that need to be associated with the Class. If an existing Class contains a subset of the desired Attributes then it is a potential Super-Class for the new Class. Clearly the Class with the largest such subset (i.e. the lowest such Class in the hierarchy) is the logical Super-Class to choose. Next it must be determined which existing Classes should be Sub-Classes of the new Class. The answer here is that existing Classes with Attributes which are a superset of the new Class should be Sub-Classes of the new Class. A tool to determine the relations between sets of Attributes is clearly suggested. This may be incorporated into the CH Navigation Tool. If a suitable super Class is not identified, the Class can be considered to represent a leaf node, a Sub-Class of the Class with the largest sub-set of desired Attributes.

Removing a Class is effectively like removing a set of Attributes although existing Sub-Classes of the Class need to become Sub-Classes of its immediate Super-Class. Problems still arise if any specialisation of Attribute Values, or addition of Attributes, were made in the removed Class. Clearly such Attributes must be re-introduced, or specialisations made, either in the Sub-Classes or the immediate Super-Class, as seems to be most appropriate.

4.3. Changes to the Source Data.

At a higher level, Rules will also be jeopardised by changes in the source material, as when legislation is amended. A feature of the MADE development methodology is that a linking facility is provided to link individual sections in the source material through the various analysis stages to the resulting CHs and RBs in the target representation. This provides a useful basis for identifying Rules and Classes that may be affected as a result of changes in the source material and can be automated as part of the Jeopardy Tool (T3).

4.4. Validation.

So far only the verification of the RB and CH have been considered. However it is also necessary to validate the KBS after maintenance has taken place. This can be carried out by peopling the Rule Base and determining what inferences can be made or tracing how inferences are made. Thus:-

T5 Rule Base Animation Tool.

5. INTRODUCTION TO PROPOSED TOOLS.

In the previous Section a number of maintenance tools were identified to address different aspects of KB maintenance. In the following Sub-Sections each of these tools will be described in greater detail.

5.1. The Rulemap (T1).

The Rulemap is a directed (from left to right) bipartite graph which graphically displays the Rule Base either at the MIR or CMIR level. A number of options are provided to allow the user to walk up and down the Rule Base. By following a path through the Rulemap it is possible to determine the Leaf

Attributes and Propositions into which a Root Attribute ultimately unfolds and vice versa. This gives the user a clear visual view of the rules in the knowledge base. Options are also provided to allow the user to interrogate the Rulemap to display the Rule and Clauses in which Attribute and Propositions appear or to inspect the Values or Entities associated with Attributes and Propositions. There is also a facility to identify redundant or subsumed Rules or sub-sets of Rules.

5.2. Hard Contradiction and Soft Inconsistency Identification (T2).

When adding or modifying rules in a KB during a maintenance session a hard contradiction may be introduced. In logical terms this means that there can be no model for the knowledge base, so the knowledge base cannot be correct. Soft inconsistency is a modified phenomenon and occurs when some Proposition is a consequence of the KB when it is in fact known that its negation is possible. This means that the knowledge base excludes some models which are known to occur, and again suggests a defect in the knowledge base. Thus a minimal validation of the knowledge base will involve ensuring that neither of these situations exist. Tools are under development in the MAKE project which will allow such inconsistencies in a knowledge base to be detected.

5.3. The Jeopardy Tool (T3).

This is a general purpose maintenance tool to identify jeopardised Slots and Rules as a result of changes to the source data or changes to the RB or CH. The tool will incorporate the following facilities:-

a) Rules Jeopardised by Slot Changes Identification. A facility to identify Rules jeopardised as a result of changes to Slots in the Class Hierarchy.

b) Class Definitions Jeopardised by Class Deletions Identification. A facility to identify the Slots Jeopardised by the removal of a Class because they are typed to that Class.

c) Rules Jeopardised by Source Changes Identification. A facility to identify the Rules that are effected by changes in the source material.

d) Slots Jeopardised by Source Changes Identification. A facility to identify the Slots in the CH that are effected by changes in the source material.

e) Rules Jeopardised by Rule Changes Identification. In a task dependent RB Rules above and below an altered Rule may be jeopardised. This facility will identify the sub-set of Rules which have been affected by a KB maintenance session.

The jeopardy tool operates using the links that should be included by the knowledge Engineer during system development. This is an essential part of the MADE methodology.

5.4. The Class Hierarchy Navigation Tool (T4).

This is a Class-Instance Browser designed to allow the user to navigate through a CH. The tool is intended to give visibility to the author of not just the CH, but also where Slot definitions come from, and will enable the user to determine the best location for new Classes, Sub-Classes and Attributes related to those Classes, and specialisations of Attribute Values.

5.5. The Rule Base Animation Tool (T5).

This tool is similar to the Rulemap but addresses the dynamic aspects of the RB. It allows the user to people the Rule Base by creating Instances and asserting Propositions and then determine what inferences can be made as a result. When the behaviour is unexpected, either because an inference which should not be made is made, or because an inference which was expected fails to be made, this tool will enable the the user to locate the precise Clause which caused the failure, and from this the Rule, analysis and source from which it was derived. Such animation is a necessary adjunct to the "by eye" validation supported by the static tools, since the practical consequences of a given fragment of the KB may be hard to envisage in the abstract.

6. CONCLUSIONS.

In this paper we have given a brief overview of KANT and KBB and identified a number of maintenance tasks that may be required for systems developed in this environment. To assist in these maintenance tasks a number of maintenance tools have been suggested and briefly described. These descriptions do not in any way represent a precise specification. They are however an indication of how the proposed tools may be implemented.

For many of the proposed tools proto-types have been built, if not the tools are at least well understood. Where proto-types have been built these have been implemented on a pilot version of the British Coal KBS which has been developed as part of the MAKE project and have been well received. It is to be hoped that all the tools described can be at least be partially built during the duration of the MAKE project. At present work is proceeding on the specification and development of the Rulemap tool.

7. ACKNOWLEDGEMENTS.

The work described here is being carried out as part of the MAKE Project, supported by the Information Engineering Directorate of the UK Department of Trade and Industry and the UK Science and Engineering Research Council. The project collaborators are ICL, the University of Liverpool and British Coal.

REFERENCES

[BEN91a]
Bench-Capon, T.J.M. & Coenen, F. (1991) *Practical Application of KBS To Law: The crucial Role of Maintenance* in van Noortwijk, Class., Schmidt, A.H.J. & Winkels, R.G.F. (Eds) *Legal Knowledge Based Systems: Aims for Research and Development.* Koninklijke Vermande BV.

[BEN91b]
Bench-Capon, T.J.M. and Forder, J.M. (1991). *Knowledge Representation for Legal Applications* in Bench-Capon, T.J.M. (ed), *Knowledge based systems for Legal Applications.* Academic Press, pp 245-264.

[COE90]
Coenen, F. and Bench-Capon, T.J.M. (1990). *Initial analysis of Maintenance Issues Associated with Knowledge Based Systems.* MAKE Report 2/90, Liverpool University.

[DAV84]
Davis, R. (1984). *Interactive Transfer of Expertise.* In Buchannan, B.G. & Shortliffe, E.H. (1984) *Rule-Based Expert Systems: The MYCIN Experiments of The Stanford Heuristic Programming Project.* Addison-Wesley.

[FOR91]
Forder, J.M. & Taylor, A.D. (1991). *The Local Office System.* in Bench-Capon, T.J.M. (ed), *Knowledge based systems for Legal Applications.* Academic Press, pp 139-164.

[PRE88]
Pressman, R.S. (1988). *Software Engineering, A Practitioners Approach, 2nd Edition.* McGraw-Hill.

[ROU88]
Rousett, M. (1988). *On The Consistency of Knowledge Bases: The COVADIS System.* Proceedings of ECAI 88.

[STO89]
Storrs, G.E. & Burton, C.P. (1989). *KANT, A Knowledge Analysis Tool.* ICL Technical Journal, Vol 6, No 3, May 1989.