--

--

KBS DEVELOPMENT USING X WINDOWS: THE MADE DEVELOPMENT METHODOLOGY

Frans Coenen, Trevor Bench-Capon

Department of Computer Science,
The University of Liverpool,
PO BOX 147, Liverpool L69 3BX.

email frans@uk.ac.liv.cs

Mick Smith

International Computers Limited,
20 St Mary's Parsonage,
Manchester M3 2NL.

ABSTRACT

This paper describes the development of a Knowledge Based System (KBS) for British Coal using the Make Authoring and Development Environment (MADE). This environment is based on X windows and is designed to take the developers from a set of textual sources through to a final application. MADE is founded on a KBS development tool called KANT (Knowledge engineers assistANT). The advantages of MADE are firstly that it is self contained in that the development environment is also the final application environment. Secondly that the philosophy behind the methodology is such as to encourage the production of KBS which are readily maintainable through an isomorphic approach to knowledge analysis and KB development and the implementation of a suite of maintenance tools. The intention of isomorphism is that the Rule Base should faithfully reflect the sources from which it was derived. The development environment is designed to be implemented on SUN work stations operating under UNIX. The system consists of three Base Windows referred to as the KANT, MADE and MAPPE (Make APPlications Environment) windows. In the KANT Base Window the knowledge analysis part of the development takes place which results in an intermediate representation of the knowledge to be displayed by the system. This representation is then compiled and will form the final executable application. The compiled MIR can be interacted with through the MADE Base Window, where various verification and validation support tools can implemented on the application. The MAPPE Base Window provides the end user view.

June 16, 1998

--

--

KBS DEVELOPMENT USING X WINDOWS: THE MADE DEVELOPMENT METHODOLOGY

Frans Coenen, Trevor Bench-Capon

Department of Computer Science,
The University of Liverpool,
PO BOX 147, Liverpool L69 3BX.

email frans@uk.ac.liv.cs

Mick Smith

International Computers Limited,
20 St Mary's Parsonage,
Manchester M3 2NL.

1. INTRODUCTION.

The development of a regulation based Knowledge Based System (KBS) using the Make Authoring and Development Environment (MADE) and methodology for British Coal (BC) is described. This was developed as part of the Maintenance Assistance for Knowledge Engineers (MAKE) project. This is a collaborative project between Liverpool University, ICL and BC directed at developing a suite of maintenance tools, suitable for use on regulation based KBS, supported by a methodology that encourages the production of maintainable systems. MADE is the result of the latter. The methodology supports KBS maintenance by basing system development on isomorphic principles.

The intention of isomorphism is that the Knowledge Base (KB) should faithfully reflect the sources from which it was derived and that the source, in turn, should be recoverable from the KB. In regulation based KBSs this is supported by the well-defined structure of the legal sources, used to develop the system, which themselves represent a multi-layered logical structures (Routen (1989)). The advantages of using an isomorphic representation pervade the whole life cycle of KBSs. It has been found that development is facilitated, validation and debugging supported, and maintenance made very much easier. Moreover isomorphism carries over into the execution of the system, enhancing the application from the users' perspective as well. The advantages to be gained are discussed further in (Bench-Capon (1991a)).

The BC application is designed to supply British Coal's Insurance and Pensions Division with decision support when determining the degree of liability to admit when considering claims for compensation by employees as a result of work related injuries. The operation of MADE is described by considering the development of a fragment of the BC application.

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 (Storrs (1989)). 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 of the analysis is a set of Rules, contained in a Rule Base, and a hierarchical Object Base, called the Class Hierarchy. The Class Hierarchy Defines the vocabulary of Object Slots and Values which can be used in constructing Rules. It also describes the state of the application at any given time. The Rule Base and the Class Hierarchy are developed in an intermediate representation called MIR (Make Intermediate Representation). This is essentially a simple language to define Objects and Rules using first order predicate logic with some extensions, for example to handle arithmetic. The nature of this intermediate representation and the advantages to be gained are fully described in ((Bench-Capon (1991b))).

The MIR Rule Base and Class Hierarchy are compiled into the target representation used in the MAKE Inference Engine (MIE). Currently this consists of a Clausal form referred to as CMIR (Compiled MIR). Other target representations could be used equally well. The MADE window is the user interface through which the MAKE maintenance tools are invoked. These all operate on the compiled version of the MIR. A full discussion of the tools available under the MAKE project is given in (Bench-Capon (1991c)).

The MAPPE Base Window provides the end user view of the application. It is organised as a hierarchy of Topics or Question forms which the user is invited to fill in. The answers to questions are 'asserted' into Object Slots as Values, which in turn trigger Rules in the Rule base. The structure of this Topic hierarchy is used to reflect the natural (task) decomposition of the problem.

3. SYSTEM DEVELOPMENT USING MADE.

System development using MADE consists of an iterative process in which repeated loops are performed, perhaps indefinitely, in that requirements will change over time and source material will be superseded. Each iteration involves the construction of four levels of (KANT) structures as follows:-

1) Source Documents:

These are imported into the KANT system, preserving their original format, ready for analysis. KANT supports the 'Copying and pasting' of words, phrases etc from these documents into other structures, and also the definition of 'Links' between such 'sources' and the new structures.

2) Freestyle KANT Structures:

These can best be described as 'Structured English' notes. KANT readily supports the structuring of such 'notes' (called Nodes) through the use of Child and Sibling relations, and through separable fields within a Node. The author is free to construct any number of such structures, with a variety of contents. However the proposed methodology strongly suggests a number of specific structures, these ae detailed later in this document.

3) MIR Structures:

These are yet more KANT structures, but constructed according to a defined syntax. They not only define Rules and Object Classes, but also the Topics and Questions to be used in the dialogue with the end user.

4) Compiled MIR:

This is an executable representation of the MIR, in that it can be interpreted by the MAKE Inference Engine (MIE), forming the dynamic, instantiated, End User Application. It is envisaged that alternative Compilers can be written in order to target for different End User Environments. Thus if Authoring and Maintenance tasks can be confined to operate on MIR Structures and preceding structures/sources, then the MADE can be used as a generalised development environment.

For Legislation Based KBS systems the proposed methodology suggests three strands of authoring material as shown in Figure 1. The Dialogue strand results in the user interface and the Object Classes and Rules strands in the Class Hierarchy and Rule Base respectively. As can be seen a number of Freestyle KANT structures are used in the analysis gradually building up to the definition of Rules, Object Classes, Topics and Questions in MIR. It is important to remember that in this entire analysis links are being constructed such that any MIR definition can be traced back to the original source material and vice versa. The method actually encourages the isomorphism between source and MIR definitions through the copy and paste operations offered by KANT. The MIR definitions are then compiled into the target representation, CMIR.

In the following Sub-Sections each stage in the development of the Class Hierarchy and Rule Base strands will be considered in further detail by considering a fragment of the BC application in terms of the

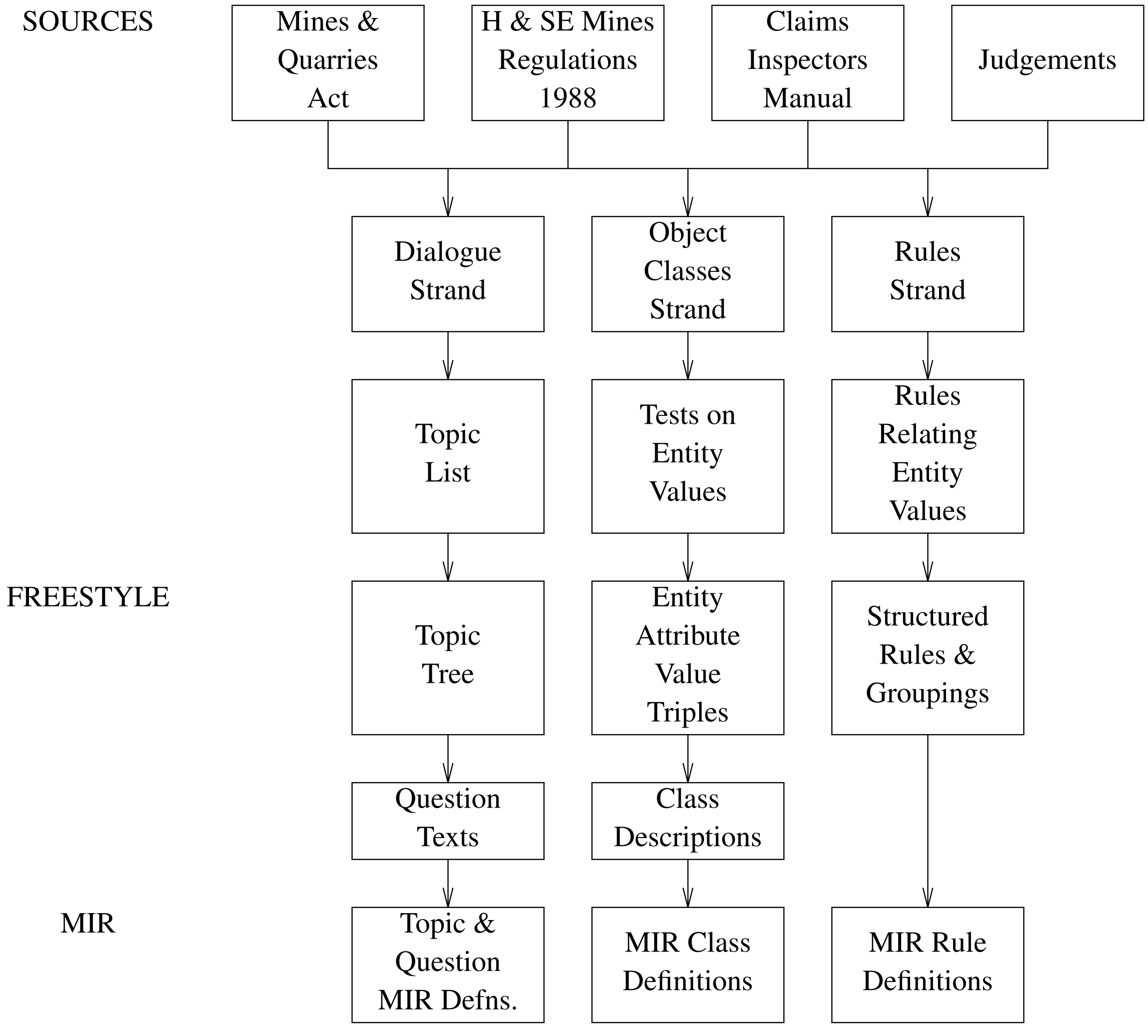

Figure 1. Strands of authoring material for a regulation based KBS.

four levels of KANT structure created on each itteration. This fragment is concerned with the safety of buildings and structures on the surface of a mine. This fragment will be used to build a sample KB, consisting of a Rule Base and Class Hierarchy, to establish whether the mine manager has complied with the Regulations or not.

3.1. IMPORTATION OF SOURCE DOCUMENTS.

Importation is the process of taking raw texts and moving them into a KANT source format but preserving its original appearance. The source documents used in the development of the BC application were:-

(i) The Mines & Quarries act 1954.

(ii) The H & SE Mines (safety of Exit) Regulations 1988.

(iii) The Claims inspectors Manual (1990).

(iv) A number of significant judgements.

If we consider the fragment of the BC application concerned with the safety of surface buildings and structures the relevant section from the Mines and Quarries Act is as follows:-

Buildings, Structures, Means of Access, &c.

86. All buildings and structures on the surface of a mine shall be kept in safe condition.

87.-- (1) There shall be provided and maintained safe means of access to every place in or on a building or structure on the surface of a mine, being a place at which any person has at any time to work.

(2) Where a person is to work at any such place as aforesaid from which he will be liable to fall a distance of more than 10 feet, then, unless the place is one which affords secure foothold and, where necessary, secure handhold, means shall be provided by fencing or otherwise for ensuring his safety.

In its imported form the above will consist of four blocks of text, the title and paragraphs 86, 87.1 and 87.2. These blocks or parts of them can then be copied into KANT Structures where further analysis can take place.

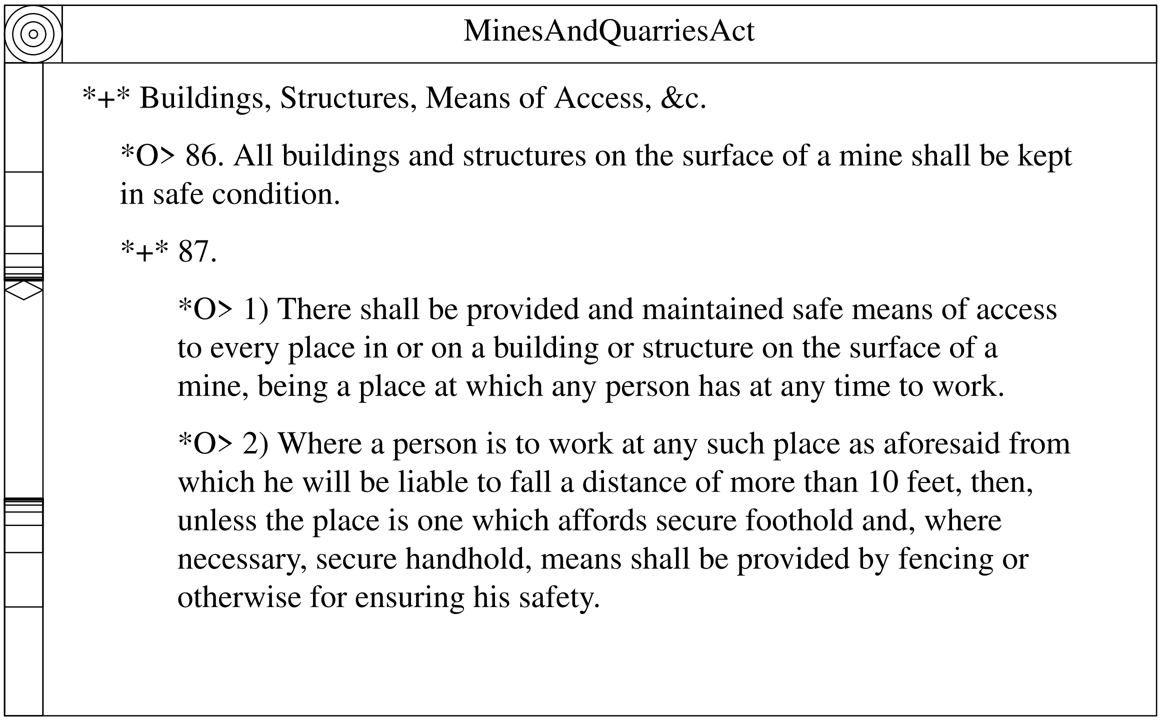

Figure 2. Source Freestyle KANT Structures.

3.2. FREESTYLE KANT STRUCTURES.

After importation knowledge analysis is commenced by examining the relevant passages in the source material to gain a broad understanding of the domain. In some cases it is advantageous to collect together the subset of Rules and regulations that are applicable to a particular part of the final application into a single KANT Structure file, using the copy and paste facilities provided with KANT. Figure 2 gives a KANT window showing the relevant legislation in Freestyle KANT Structure format. Note that at the same time links will be created between the nodes and the source. At any time the author may follow these links back to the source or, in time, forward to the Rules and objects in the final application.

It is suggested that analysis be implemented by following the steps listed below.

1) Pass through the source identifying Attributes and their entities and the tests applied to them.

2) Collect the Attributes and tests of each Entity into a Freestyle KANT Structure.

3) Collect Entity Attributes Value triples (EAVs) together and form another Freestyle KANT Structure.

If we consider the above imported Regulations we can identify three principal entities, buildings and structures on the surface, places where persons work and high places where persons work, i.e. buildings, work places and high work places. It is important to appreciate that high work places are a type of work place and that work places are a type of building or structure. We can also identify a fourth implied Entity, this is the mine manager who is responsible for ensuring that the Regulations are complied with. The Entity mine manager can be considered to be a type of manager which in turn is a type of person. We could identify further Entities such as means of access, foothold and handhold. However within the example fragment of legislation no test are carried out on these Entities and so they can be considered to be not relevant. There is also a question of grain size to be considered here. For example we may consider buildings and structures to be two separate Entities. However in the piece of legislation under consideration here no distinction is made between the two and thus they can effectively be considered to be the same thing.

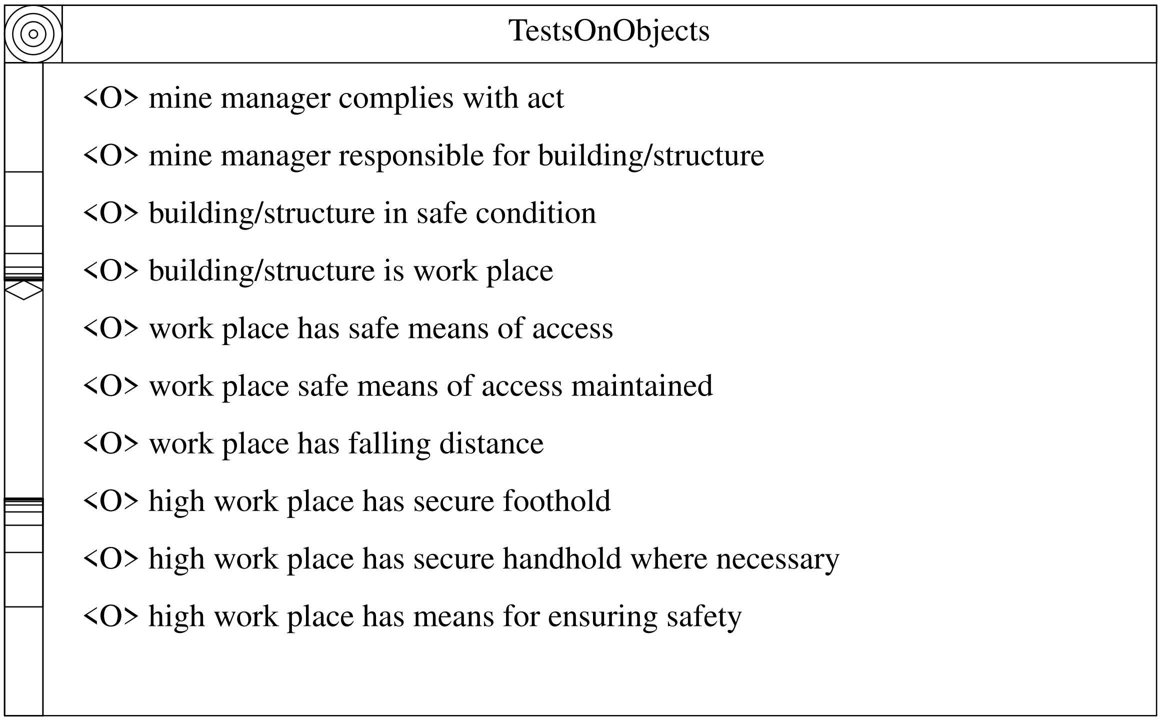

Figure 3. Tests On Objects Freestyle KANT Structures

Having identified the relevant Entities contained in the Regulations we can consider the tests that may be performed on them. For example from Section 86 of the Regulations we can identify a test to determine whether a building/structure is in a safe condition. Similarly from Section 87.1 we can identify a test to determine whether a building/structure is a work place. At a higher level we can presume the existence of a test to determine whether a manager complies with the act or whether he/she is responsible for the building/structure in question. All relevant tests can be collected into a single KANT structure file and displayed in a KANT window as shown in Figure 3.

We are now in a position to consider EAVs. We can identify the Attributes associated with the Entities and the Values those Attributes can have by inspection of the Tests On Objects and the source. Firstly, from the Tests On Objects file, we can extract an Attribute for the Entity mine manager that the manager complies with act and that this Attribute can have the Value True or False. Alternatively if we consider the Entity building/structure we can identify further Attributes and Values. For example the Attribute in safe condition which can have the Value true or false, or is work place which can also have the values true or

EAVs

<O> manager - complies with act - true, false

<O> manager - responsible for - building/structure

<O> building/structure - in safe condition - true, false

<O> building/structure - is work place - true, false

<O> work place - complies with work placed regs - true, false

<O> work place - has safe means of access - true, false

<O> work place - safe means of access maintained - true, false

<O> work place - has falling distance - 0 to big

<O> high work place - complies with high work place regs - true, false

<O> high work place - has secure foothold - true, false

<O> high work place - has secure handhold where necessary - true, false

<O> high work place - has means for ensuring safety - true, false

Figure 4. EAV Freestyle KANT Structures

false. The EAV triples identified are stored in another KANT structure file of the form shown in Figure 4.

In the above EAV and Tests On Object structures care has been take to ensure that each structure or sub-structure bears a direct, one-to-one, correspondence with the source material i.e. isomorphism with the source is maintained. Further note that the above analysis stages are designed to minimise and contain the need for redoing the analysis as a result of problems located at later steps. It is recommended that the author uses source material terminology for Entity, Attribute and Value names wherever possible.

3.3. MIR STRUCTURES.

In the Intermediate Representation stage the Tests On Objects and EAV structures are developed into a MIR Class Hierarchy and Class Hierarchy. In this case the Classes and Rules are stored in MIR KANT structures.

Considering the Class hierarchy first this is created by collecting the Attributes and Tests for each Entity and defining the Attribute set for each Entity. At the same time checks should be made for duplication and overlap. The Entities are collected together in a Class Lattice, common Attributes are placed up to their highest common level. The result is a formal definition of Entities, Attributes and Types in the MIR language. For the fragment of KB under consideration here the Class Hierarchy in MIR will be of the form given in Figure 5. Note that the Class Hierarchy is arranged so that Sub-Classes inherit Attributes from Super-Classes. For example a HighWorkPlace inherits the Attribute keptInSafeCondition.

With respect to the Rule Base we can pass through the Sources, Tests On Objects and EAV structures again and identify Rules. Thus we can construct a Rule to establish whether the mine manager complies with the Act:-

```
ClassHierarchy

*+* TopObject

    <O* MineManager
    ^CompliesWithAct (true false)
    ^ResponsibleFor - BuildingStructure

    *+* BuildingStructure
    ^CompliesWithWorkPlaceRegs (true false)
    ^InSafeCondition (true false)
    ^IsWorkPlace (true false)

        *+* WorkPlace
        ^HasSafeMeansOfAccess (true false)
        ^SafeMeansOfAccessMaintained (true false)
        ^HasFallingDistance (0  big)

            <O* HighWorkPlace
            ^CompliesWithHighWorkPlaceRegs
            ^HasSecureFoothold (true false)
            ^HasSecureHandhold (true false)
            ^HasMeansForEnsuringSafety (true false)
```

Figure 5. Class Hierarchy MIR KANT Structure.

```
(manager CompliesWithAct is true
iff
(((BuildingStructure InSafeCondition is true)
and
((BuildingStructure IsWorkPlace is false)
or
(BuildingStructure CompliesWithWorkPlaceRegs is true)))
when
(manager Responsible BuildingStructure)))
```

3.4. COMPILED MIR.

During the compilation stage the MIR Class Hierarchy and Rule Base are compiled into Objects and Rules in the executable target representation, CMIR. In the case of the Rule Base this will result in a set of Clauses. If we consider the Rule constructed in the previous Section this will compile into the following Clauses.

(manager CompliesWithAct is true
iff
((BuildingStructure InSafeCondition is true)
and
(BuildingStructure IsWorkPlace is false)
and
(manager Responsible BuildingStructure)))

(manager CompliesWithAct is true
iff
((BuildingStructure InSafeCondition is true)
and
(BuildingStructure CompliesWithWorkPlaceRegs is true)
and
(manager Responsible BuildingStructure)))

(not manager CompliesWithAct is true
iff
((not BuildingStructure InSafeCondition is true)
and
(manager Responsible BuildingStructure)))

(not manager CompliesWithAct is true
iff
((not BuildingStructure IsWorkPlace is false)
and
(not BuildingStructure CompliesWithWorkPlaceRegs is true)
and
(manager Responsible BuildingStructure)))

4. CONCLUSIONS.

The MAKE Application Development Environment and Methodology have been described in terms of

- The way in which it preserves an isomorphism between source material and, ultimately, the end user application.

- The use of the MAKE Intermediate Representation as a general re-targetable step before delivery in a specific language or environment.

- An authoring process, supported by Freestyle KANT structures, which exploits audit trail information, and thus encourages the preservation of the aforementioned isomorphism.

- The linking of structures to form an audit trail which will later be used in various maintenance activities ((Bench-Capon (1991c))).

- The heavy use of graphical presentation of the material produced.

It is argued that all these features put together, as in the MADE system, combine to make a very practical development environment for regulation based KBSs which addresses the whole application life cycle.

5. ACKNOWLEDGEMENTS.

The work described above was 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. In addition the authors would like to express special thanks to Charlie Portman for his working paper "Authoring im MAKE."

REFERENCES.

1. Bench-Capon, T.J.M. & Coenen, F.P. (1991), "Exploiting Isomorphism: Development of a KBS to Support British Coal Insurance Claims", Proceedings of the Fourth International Conference on AI and Law, Oxford. ACM Press, 1991.

2. Bench-Capon, T.J.M. & Forder, J.M. (1991), "Knowledge Representation for Legal Applications", in Bench-Capon, T.J.M. (Ed), "Knowledge Based Systems and Legal Applications"' Academic Press, pp 245-264

3. Bench-Capon, T.J.M and Coenen, F.P. (1991), "Practical Application of KBS to Law: The Crucial Role of Maintenance", in van Noortwijk, C., Schmidt, A.H.J., Winkels, R.G.F. (1991). "Legal Knowledge Based Systems, Aims for Research and Development", Koninklijke Vermande BV, Lelystadt, Netherlands. pp 5-17.

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

5. Routen, T. (1989), "Hierarchically Organised Formalisations", Proceedings of the Second International Conference on AI and Law, Vancouver. ACM Press, 1989.